Querying and Visualizing Gridded Datasets for e-Science

Bill Howe
OGI School of Science & Engineering at
Oregon Health & Science University
Beaverton, Oregon
bill@cse.ogi.edu

David Maier
OGI School of Science & Engineering at
Oregon Health & Science University
Beaverton, Oregon
maier@cse.ogi.edu

Abstract

We demonstrate a web service and client application for querying and visualizing datasets defined over arbitrary topological grids. Such gridded datasets are produced by Earth science simulations, 2D and 3D visualization software, and image processing systems. Existing systems for manipulating gridded datasets capture only simple grids or do not support remote access. We exercise a novel data model by using it to build a web-enabled visualization system. We show that complex visualizations, including animations, can be expressed using our data model and associated algebra. Further, we demonstrate a standards-based, service-oriented architecture for processing the algebra expressions and returning results over the web.

1. Introduction

Transparent sharing of scientific data is becoming a primary requirement for the Earth science community [2]. We have developed a mechanism for sharing a particular class of scientific datasets: those defined over a topological *grid* structure[1]. For example, a timeseries might be defined over a 1-dimensional grid, while fluid flow over an airplane wing might be defined over a 3-dimensional grid.

Gridded datasets are difficult to accommodate using existing database technology due, in part, to data model limitations [8, 9]. In recent work [8], we presented a novel algebra of *gridfields* for manipulating gridded datasets. In this demonstration, we show how our algebra can be used to enable remote processing of gridded datasets, which in turn facilitates data sharing between scientific communities.

The context for our interest in gridded datasets is CORIE [4], an Environmental Observation and Forecasting System designed to support scientific and industrial interests in the Columbia River estuary. The CORIE system both measures and simulates the physical properties of the estuary, generating 5GB of data and thousands of *data products* for each simulation run, including visualizations, aggregated results and derived datasets. The data products are consumed for many purposes, including salmon habitability studies and environmental impact assessments.

In the current production CORIE system, "canned" visualizations are produced eagerly for every run. These visualizations are organized by date and published on the web. Users cannot customize their data products nor access the data directly.

Only a few use cases are satisfied by this rigid infrastructure. Many potential users require access to the data itself. Weather forecasters can use CORIE results as a regional inputs for global simulations. Government agencies can investigate much smaller scale phenomena, incorporating salinity data into, e.g., groundwater simulations along the estuary's shore. Colleagues who collect or generate data over the same domain can compare results with the CORIE system quantitatively rather than relying on subjective interpretation of gif images. Using our web service, clients can not only retrieve data, but they can perform arbitrary transformations on the data prior to download. Using our prototype client application, these datasets can be visualized automatically (Figure 1), i.e., without specifying parameters.

Sharing scientific data with other researchers is certainly not a goal unique to the CORIE system. The term e-Science is used to connote global, distributed collaboration enabled by sharing of both data and compute resources [1].

1 Our usage of the term "grid" is not directly related to the notion of Grid computing. Unfortunately, both meanings are used in the context of e-Science. We use the term "grid" for consistency with our previous work, but the word "mesh" also suffices.

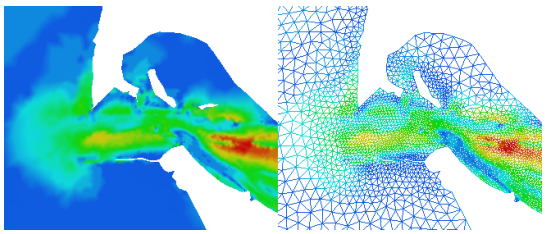

Figure 1. A visualization of the water's surface, shaded by the magnitude of horizontal velocity.

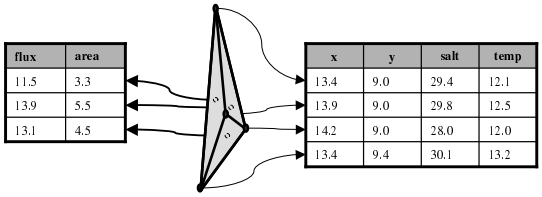

Figure 2. Datasets bound to the nodes and polygons of a 2-dimensional unstructured grid.

The systems to support e-Science are still evolving. Advocates of Grid computing [7] suggest that large-scale scientific data management requires seamless inter-operation of heterogeneous compute resources and heterogeneous data sources. Currently, these *Data Grid* technologies are in their infancy, adopting uninterpreted files as the currency of exchange and analysis [3]. As the database community has demonstrated, relying on client applications to interpret raw files leads to a problem of *data dependence*, where client applications are brittle with respect to changes in data location, organization, or access methods.

Perhaps the more direct cost to file-based data sharing is performance. Clients must download and process entire files rather than working with only their particular region of interest. Further, coarse-grained processing of entire files impedes parallelization. Fine grained operations over a logical model can be scheduled to different nodes to improve utilization.

The relational model solved the data dependence problem for table-structured data by providing a level of indirection between applications and physical file organization. Applications interact with a logical view of the data, while the database server software manages the physical realities of files, indexes, and access paths.

A similar approach is appropriate for grid-structured data. To illustrate our approach, we demonstrate a web service for processing queries remotely over gridded datasets, and a client application for visualizing the results. Before we describe these components, we briefly review the data model presented more thoroughly in previous work [8].

2. Data Model

A *node* is a named but otherwise featureless object. A *cell* is a sequence of nodes; a polygon in 2 dimensions, or a polyhedron in 3 dimensions. A *grid* is a collection of cells organized by their dimension. Unlike spatial database systems, nodes are referred to by an internal identifier rather than a set of geometric coordinates. With node ids, we can reason about topological properties of grids independently of any particular geometric embedding. For example, we can reason about "neighborhoods" of cells without knowledge of (x, y, z) coordinates of their nodes. Therefore, we can answer questions such as "Are these two cells touching?" without invoking geometric algorithms. These features are important for efficient processing [8].

A *gridfield* consists of a grid and a dataset bound to the cells of a particular dimension. We might bind timestamps to a 1-dimensional gridfield to construct an ordered sequence with a "linear" topology. In the CORIE system, examples of bound datasets include the physical characteristics of the ocean and river, as well as simple x, y, z coordinates.

Figure 2 shows a 2-dimensional unstructured (non-rectilinear) grid with two datasets bound to it. Geometric coordinates x and y are bound to each node of the grid, as are salinity and temperature values. Area and flux values are bound to each 2-dimensional polygon.

We have also defined several operators over gridfields. The restrict operator is analogous to the relational select. The bind operator reads in an array and associates it with an existing grid. The merge operator computes the intersection of two grids, and rebinds the data appropriately. The cross product operator computes an d-dimensional gridfield from two gridfields of dimension x and y, where $d = x + y$. To visualize a product grid, consider that the cartesian plane can be viewed as the cross product of two real number lines. If those two number lines are divided into discrete segments, then their cross product will be divided into discrete rectangles. In our model, we formalize and generalize this intuition to handle arbitrary grids.

The most expressive operator in our algebra is aggregate. The aggregate operator can be used to map one grid onto another, apply a function to the bound data, smooth noisy data by averaging over neighborhoods, or aggregate over the entire grid to produce a

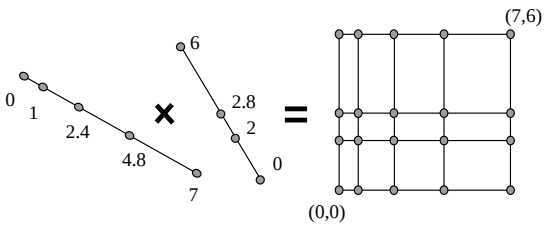

Figure 3. Constructing a 2-dimensional regular grid from 1-dimensional regular grids. Many data models support regular grids only.

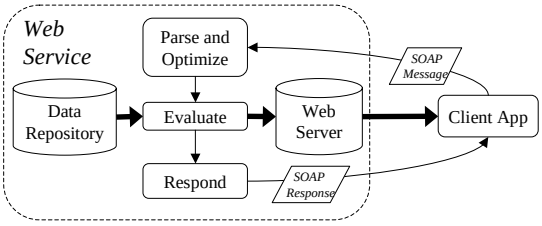

Figure 4. Architecture of the web service and client application.

single value. The operator is programmed by two functions: an *assignment* function to map cells in the source grid to cells in the target grid, and an *aggregation* function to transform the data values bound to the mapped cells.

Many existing data models only support *regular* or *rectilinear* grids. To illustrate that these grids are a subset of the grids we can express in our model, consider Figure 3. A *linear* 1-dimensional grid is a sequence of nodes connected by 1-dimensional edges. A *regular* grid is a linear grid or the cross product of two regular grids. Regular grids are isomorphic to multidimensional arrays, where the linear grids represent the individual dimensions. Rectilinear gridfields can be constructed similarly by binding coordinate values to the individual dimensions. In Figure 3, a sorted array of x-values is bound to one linear grid, and a sorted array of y-values is bound to another linear grid. The cross-product of the two linear grids results in a 2-dimensional rectilinear grid representing the xy plane.

Existing data models targeted at gridded datasets (e.g., [5]) capture only regular or rectilinear grids. The gridfield in Figure 2 is unstructured, meaning that it contains no regularity to simplify processing; this gridfield cannot be expressed in most data processing systems. Visualization libraries can manipulate unstructured grids, but do not offer an algebra to simplify the semantics and afford optimization. Further, visualization systems do not provide means of processing unstructured grids via loosely-coupled web services.

Another feature of our data model is *nested* gridfields. In a nested gridfield, attributes associated with cells need not be numeric values, but can themselves be gridfields. Nested gridfields can be used to model multiresolution datasets or any other hierarchical structure. In this demonstration, we use nested gridfields to organize the frames of an animation, and to spatially arrange multiple simultaneous visualization on screen.

3. Architecture

Both the server and the client are written in c++, primarily for efficiency. Individual operators on the server are compiled separately and linked via materialized intermediate results. Obviously, this approach incurs additional IO costs, but it also allows independent operations to run on different nodes on a cluster of computers. We defer sophisticated execution strategies for future work.

Server. Our server is implemented as a web service, responding to XML requests and returning XML responses with a URL to an associated binary payload. Scientific datasets are usually too large to be translated into any ASCII representation, and schemes for transporting binary data directly within an XML document are fraught with problems [6]. Instead, we have chosen to use the Simple Object Access Protocol (SOAP) [10] for passing messages, but use a separate protocol to transfer the potentially large query results.

Figure 4 illustrates our architecture. Thick arrows represent flow of (large) gridded datasets, thin arrows represent flow of (small) query plans or control information. A SOAP request is received from the client and parsed. Some rudimentary algebraic optimizations are applied, such as pushing restrictions down the tree. Then the plan is executed, using pre-compiled query operators. The output of the query is materialized and exposed to the Internet. The URL is sent back to the client, enveloped in a standard SOAP response.

Providing only indirect access to the query results opens up some interesting processing strategies. Clients might ask for the results to be computed asynchronously, accessing them at a later time. Further, clients could track URLs for computed results and serve this information to other sub-clients. Obviously, the lifetime of computed query results cannot be indefinite, but waiting a day or more before purging them is not infeasible at our current usage. Developing full-featured strategies for pre-computing and caching results is beyond the scope of this work.

GridField Structure	Visualization Dimension
"x", "y", and "z" attributes	Spatial Dimensions
1 non-spatial attribute	Color
2 or 3 non-spatial attributes	Vector Glyphs
4+ non-spatial attributes	Glyphs and Color
Nested Linear Grid	Animation
Nested 2-dimensional Grid	Screen Layout
2+ innner grids in one cell	Superimposition

Table 1. Mapping gridfield structure to visualization techniques.

Client. The client application generates visualizations automatically based on the structure of the gridded dataset being viewed. Visualization systems offer only a few core dimensions onto which we can map data. Three spatial dimensions can be visualized, a fourth dimension can be captured via animation, visual objects can be colored by a scalar quantity, and glyphs can be used to visualize 2 or 3 dimensions of vector data. Tuning these dimensions to convey maximum information to the viewer is a separate area of research; we just adopt a few simple conventions that are useful in practice. Table 1 shows how gridfield structures are mapped to visualization dimensions. Attribute names specify spatial dimensions, and extra attributes are interpreted as scalar or vector quantities depending on how many there are. A linear outer grid is interpreted as representing time; each inner grid is rendered as a frame of an animation. A 2-dimensional outer grid allows control of the arrangement of the inner grids on the screen. Figure 1 illustrates an example of this interpretation. Finally, multiple gridfields can be displayed in the same visualization if each are associated with the same cell in an outer grid.

4. Demonstration Specifics

In our demonstration, we will show a form-based interface for expressing queries useful in our domain. The specific queries to be demonstrated and their associated visualizations are described below.

3-D Plume Animation. The plume is the jet of fresh water ejected from the mouth of the river. Plume dynamics are important for salmon habitability studies, commercial fisheries, and basic environmental physics. The query constructs a nested gridfield: The outer gridfield is simply time, and the inner gridfield is a 3-D representation of the plume. Each frame of the animation displays the portion of the grid where the salinity attribute is below a given threshold. The plume is positioned in space by attributes representing x, y, and z coordinates, and the coloration is determined by the salinity attribute.

2-D Vertical Profile, Two Windows. This data product displays a vertical profile of the river. The top and bottom of the image correspond to the surface and bottom of the river. We use two layers of nesting here: one for the animation, and another for the arrangement of two windows, each showing a different variable over the same grid. This query uses a technique to lower the dimension of the intermediate results, and thereby improve performance [8].

3-D Surface Animation. This data product displays a 2-D manifold representing the surface of the estuary, warping it in space as the tide ebbs and flows. Glyphs show velocity vectors, and color shows elevation, further emphasizing the vertical motion.

References

[1] R. Allan. Building the e-Science Grid in the UK: Middleware, applications and tools deployed at level 2. In *Proceedings of the UK e-Science All Hands Meeting*, September 2003.

[2] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Lee, A. Sim, A. Shoshani, B. Drach, and D. Williams. High-performance remote access to climate simulation data: a challenge problem for data grid technologies. In *Supercomputing*, pages 46–46. ACM Press, 2001.

[3] W. Allcock. *GridFTP Protocol Specification*, 2003. http://www-fp.mcs.anl.gov/dsl/GridFTP-Protocol-RFC-Draft.pdf.

[4] A. Baptista, M. Wilkin, P. Pearson, P. Turner, C. McCandlish, and P. Barrett. Coastal and estuarine forecast systems: A multi-purpose infrastructure for the columbia river. *Earth System Monitor, NOAA*, 9(3), 1999.

[5] P. Baumann. A database array algebra for spatio-temporal data and beyond. In *Next Generation Information Technologies and Systems*, pages 76–93, 1999.

[6] A. Bosworth, D. Box, M. Gudgin, M. Nottingham, D. Orchard, and J. Schlimmer. Xml, soap, and binary data. Technical report, BEA Systems and Microsoft, February 2003. www.xml.com/pub/a/2003/02/26/binaryxml.html.

[7] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.

[8] B. Howe and D. Maier. Algebraic manipulation of scientific datasets. In *VLDB*, 2004.

[9] R. Musick and T. Critchlow. Practical lessons in supporting large-scale computational science. *SIGMOD Record*, 28(4):49–57, 1999.

[10] W3C. *Simple Object Access Protocol (SOAP) 1.2*, 2003. http://www.w3c.org/TR/SOAP.