Database Support for Multisource Multiresolution Scientific Data

R. Daniel Bergeron
Ted M. Sparr

Philip Rhodes
Andy Foulks
Xuan Tang
Li Ye
Lorna Ellis
(and others)

Problem

- Scientists are faced with increasingly large and complex data sets
- Tools for managing this data are inadequate
  - scientific database systems have yet to prove useful
  - vast majority of scientists still organize data in files
- Need better support for scientific data processing
  - a formal model for describing scientific data
  - database and other software to implement that model in an efficient manner

Talk Overview

- Nature and structure of modern scientific data
- Multiresolution and adaptive resolution representations of large data sets
- Lack of good database tools for supporting scientific applications
- Formal data model for scientific data
- Prototype system to support the data model

Modern Scientific Data

- Huge in size
- Complex
  - Multidimensional and multivariate
  - Multisource
  - Distributed

Data is too large and too complex to access directly as a single entity - especially in an environment.

Scientific Data Size

- Increasing computing power means ever more *simulation* data
- Better instrumentation means ever more *sampled* data from real world phenomena
- Analyzing and understanding this massive amount of data is yet another problem, especially when humans must be involved.
- Need to reduce size to manageable levels: *multiresolution data representation*

Scientific Data Complexity

- Scientific data is usually defined in a *multidimensional space* and has multiple data values at points in that space (*multivariate*)
- Scientists often focus on small *subsets* of a very large data set, both spatially and by variate
- Scientific data is often organized in multiple sources that should be processed as a single entity
- Increasingly, scientific data is *distributed* over multiple locations

Physical Data Storage Options

- Given multivariate data at points in space
  - *point-order* storage groups the variates of each point into a record and stores each record as a unit in a file
  - *attribute-order* storage segregates all values of each variate together
- In both cases, data can be stored in multiple files
  - attribute-order data usually has 1 attribute per file
  - point-order data may be organized in *blocks* where each (spatial) block of data is stored in a separate file

Conceptual View vs. Physical Storage

- Scientist (application code) would like to view the data in a form that is natural for the task.
- Examples
  - data stored in 4 files in attribute order; program accesses it as 1 file in point order.
  - volume data stored in 100 files, one per slice; program accesses at a single 3D file.
  - 4 attribute data stored in 1 file in point order; program sees 2 attribute data in attribute order

Distributed Scientific Data

- Scientists need to access large distributed scientific data sets
- Distribution and multiresolution are natural fit
  - coarsest resolutions on workstation
  - next few finer resolutions on LAN
  - finest resolutions in archives on WAN
- Distribution and multisource data also fit
  - Spatial/temporal blocks can be distributed
  - Multiple attributes can be distributed

Distributed Scientific Environment

- Support for distributed data and processing needed by scientific applications
- *transparent* access
  - requires no special code or knowledge except the data set name (e.g., url)
- *semi-transparent* access
  - makes some aspects of the distribution visible
- *Grid computing* research should help *if* it supports small scale grids as well as super grids

Interactive Data Access Paradigm

- A scientist can only handle a small subset of the data, both computationally and cognitively
  - a low resolution abstraction of a large amount of data, *or*
  - a small amount of high resolution data
- Key requirements for interactive data access
  - *seamless transition* between resolution levels
  - *error representation* for each resolution
  - *adaptive resolution* data modelling and visualization tools

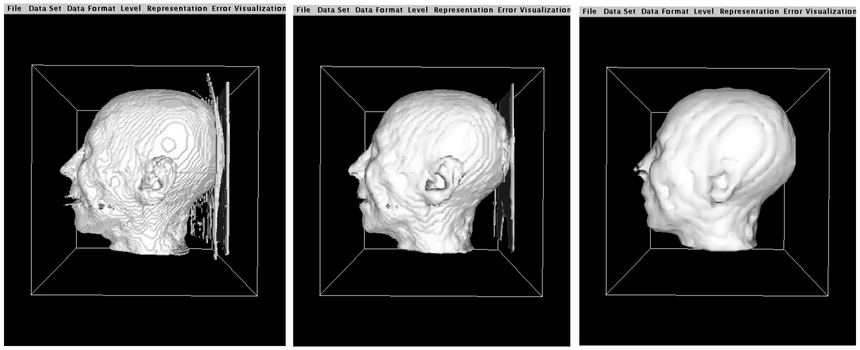
Multiresolution Data

- Hierarchy of resolutions for the same data set, generated offline
- Maintain *local error* for each resolution
- Need not save all resolution levels
- For a specific task, access the appropriate resolution level in the hierarchy, possibly determined by a user-specified *error tolerance*

Example MR Data

- Isosurface rendering of CT data (UNC data)

128x128x128 64x64x64 32x32x32



Generating MR Hierarchies

- Common ways to generate lower resolution representations from higher resolution data:
  - wavelet
  - regular or irregular sampling
  - geometry-driven simplification, such as for surfaces
- Need to produce error associated with a coarse resolution representation

Wavelets

- Wavelets provide a powerful tool for multiresolution data generation
- 1-dimensional wavelet transformation
  - N data values map to N/2 *summary* values and N/2 *detail* values
  - summary data is the lower resolution representation
  - detail data is the "error"
  - lossless transformation: can reconstruct the original data from the summary and associated detail

Haar Wavelet

- Simplest wavelet transform
- In 1 dimension
  - Given V_i, i=1...2n
  - summary: $S_j = ½ * (V_{2j-1}+V_{2j})$ average of pairs
  - detail: $D_j = ½ * (V_{2j-1} - V_{2j})$ average of differences

Haar Wavelet Example

Initial data: 4 6 9 5 8 6 3 9

Haar Wavelet Example

Initial data: 4 6 9 5 8 6 3 9

summary:

+	+	+	+
5	7	7	6
S_1	S_2	S_3	S_4

Haar Wavelet Example

Initial data: 4 6 9 5 8 6 3 9

detail:

-		-		-		-	
5	-1	7	2	7	1	6	-3
S_1	D_1	S_2	D_2	S_3	D_3	S_4	D_4

Haar Wavelet Example

Transformed data 5 -1 7 2 7 1 6 -3

group S terms:

5	7	7	6
S_1	S_2	S_3	S_4

Haar Wavelet Example

Transformed data 5 -1 7 2 7 1 6 -3

group D terms:

5	7	7	6	-1	2	1	-3
S_1	S_2	S_3	S_4	D_1	D_2	D_3	D_4

Wavelet Reconstruction

- Reconstruction builds higher resolution data from lower resolution summary and detail
- *Lossless* (if ignore numerical round-off error)
- For Haar wavelet

  Let V_j be j-th data value at higher resolution
  Let S_i be i-th summary at lower resolution
  Let D_i be i-th detail at lower resolution

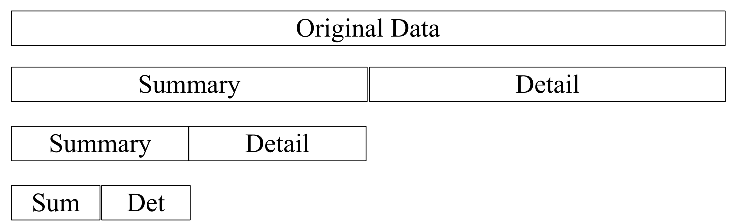
  $V_{2k} = S_k+D_k \quad V_{2k+1} = S_k-D_k$

  Verify from: $S_k = ½ (V_{2k}+V_{2k+1})$ and $D_k = ½ (V_{2k}-V_{2k+1})$

Wavelet Compression

- The summary and detail coefficients occupy as much space as the original data
- To get space savings, discard some coefficients
  - all coefficients less than a threshold
    - small summary and detail coefficients treated as 0
    - added overhead to save *position* of remaining coefficients
  - all detail coefficients less than a threshold
    - detail coefficients likely to be smaller, no position overhead needed for summary coefficients
  - all detail coefficients
    - simple storage, more error

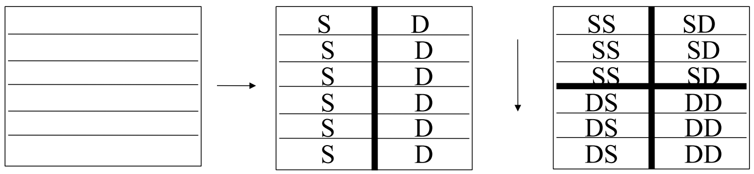
Multiresolution 1-D Wavelet



Each summary is a coarser representation of previous. Can reconstruct higher resolution from lower resolution summary and detail. If ignore detail, can reconstruct higher resolution from lower summary (assuming 0 for detail).

2-D Wavelet

- Given a 2-D array of input data
  - apply 1-D wavelet to each row
  - apply 1-D wavelet to resulting columns



- The summary data is ¼ the size of the input
- This approach extends easily to higher dimensions

Data Selection

- Simpler abstraction approach: select a *subset* of the data for the lower resolution
  - random or pseudo random selection might reduce aliasing artifacts
  - regular selection maintains distribution of data
    - discard every other data value in each dimension
    - same data reduction characteristics as wavelets
- Need to explicitly compute an <u>error metric</u>

Using Low Resolution Data

- Typically low resolution data can be used by the same algorithms that process full resolution data
- But, are the results reliable?
  - for visualization, should provide some feedback *locally* to show the authenticity of the visualization
  - Best: a single visualization including both data and error. This is hard.
  - OK: 2 renderings: one of data, one of error

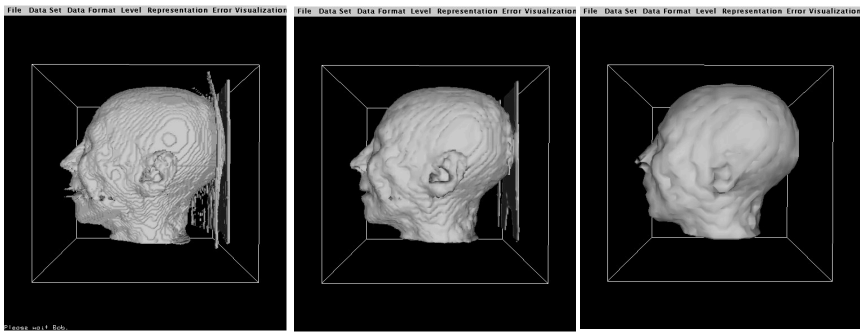
Error Generation

- Want *local error* for low resolution data
  - every data value should have corresponding error value
  - cumulative error: error resulting if low resolution data is used to reconstruct an approximation to original data
- Wavelet transform produces error for one step, but accumulating it is complicated

Example of Error Visualization

- Error is mapped to hue (red is high error)

128x128x128 64x64x64 32x32x32



Reconstruction Process

- Often need to reconstruct a higher resolution data set from a lower resolution one
- Wavelet
  - use summary and detail (which may be 0)
- Otherwise
  - Replicate low resolution points
  - Interpolate between low resolution points

Reconstruction Example 1D

- Simple 1D example

Orig data:	4	6	9	5	8	6	3	9
Summary:	5		7		7		6	

Reconstruction Example 1D

- Simple 1D example

Orig data:	4	6	9	5	8	6	3	9
Summary:	5		7		7		6	
Replication:	5	5	7	7	7	7	6	6

Reconstruction Example 1D

- Simple 1D example

Orig data:	4	6	9	5	8	6	3	9
Summary:	5		7		7		6	
Replication:	5	5	7	7	7	7	6	6
Error:	1	-1	-2	2	-1	1	3	-3

Reconstruction Example 1D

- Simple 1D example

Orig data:	4	6	9	5	8	6	3	9
Summary:	5		7		7		6	
Replication:	5	5	7	7	7	7	6	6
Error:	1	-1	-2	2	-1	1	3	-3
Interpolation:	5	6	7	7	7	6.5	6	6

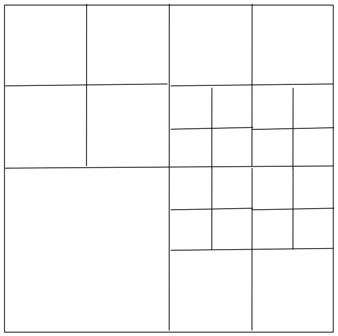
Reconstruction Example 1D

- Simple 1D example

Orig data:	4	6	9	5	8	6	3	9
Summary:	5		7		7		6	
Replication:	5	5	7	7	7	7	6	6
Error:	1	-1	-2	2	-1	1	3	-3
Interpolation:	5	6	7	7	7	6.5	6	6
Error:	1	0	-2	2	-1	0.5	3	-3

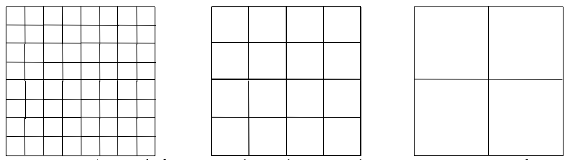
Adaptive Resolution Data

- Different resolutions in one data set
- Build by selecting parts from different MR levels
  - local resolution based on local error at chosen level
  - data storage more complex, not a simple array
- Need special analysis and visualization functions

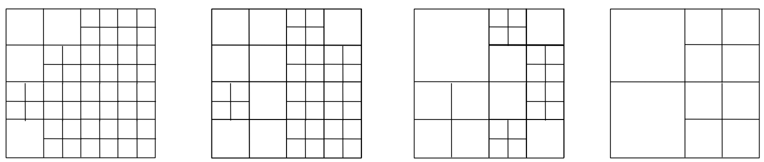


Adaptive Resolution Hierarchy

- Given multiresolution hierarchy



- Generate AR hierarchy based on error tolerances



Error: <5% <10% <15% <20%

Where are we now?

- Scientific data is huge, complex, distributed
  - What kind of support is available?
- Some reasonable *file-based packages* exist
  - Generic: CDF, netCDF, HDF
  - Many tailored to a particular application domain
  - All force user program to conform to file format
- Scientific database systems
  - aren't really a factor yet

Scientific Database Support

- So far, database support for scientific data has been very limited.
- A useful scientific database should
  - handle huge datasets efficiently
  - implement sophisticated spatial and temporal relationships
  - support interactive data exploration
  - define a comprehensive *data model for scientific data*

Data Model for MR Scientific Data

- Scientific data support packages are generally *ad hoc*
- Scientific databases have not yet proved efficient enough or powerful enough
- Need a rigorous *formal model* to provide a framework for building software support
- Need an *implementation model* that utilizes database technology where effective

Data Models

- Data models are the basis for any DBMS
- *Relational* and *object* models are most prevalent
- Neither is particularly effective for scientific data

Goals of a Scientific Data Model

- Represent a wide range of scientific data
- Model spatial relationships implicitly
- Model both multiresolution and adaptive resolution data
- Model partitioned and distributed data
- Model error in the data

Should lead to better analysis and visualization tools with less need for *ad ho*c design and/or code

Scientific Data Model

- Need to model wide range of data
  - generalize notion of a grid
  - support both spatial and temporal data
- Need formal abstractions for
  - the *domain* in which data exists
  - the *structure* of the data
  - representations of the data at different *resolutions*
  - *error*
  - *conceptual view* that differs from physical organization

Motivation for Our Data Model

- Basic MR/AR concepts are generic
- Expect to be able to make better and more general analysis and support software based on a clean and comprehensive data model – less *ad hoc* code
- Want database system support for
  - access to different resolutions
  - error representation
  - data distribution
  - processing distribution

Database Framework

- Concentrate on scientific data that is spatial (or can be treated as if it is spatial*)*
  - we call this *dimensional* data
- Database support is a critical component
  - store extensive *metadata* about the scientific data
  - access to data is via the database
- Scientific data itself is not stored in the database
  - data is in files or on the net

Key Components

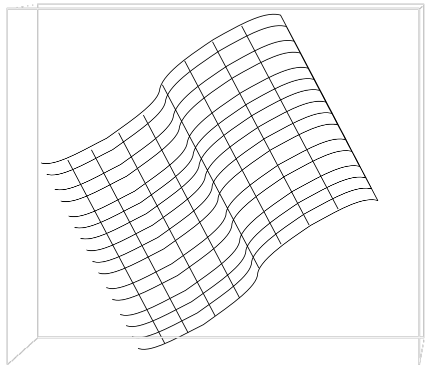
- *Lattice*
  - represents a conceptual data set
- *Topology*
  - encapsulates implicit spatial/temporal data relations
- Geometry
  - formalizes the space in which the data lives
- *DataSource*
  - abstraction of the physical storage of lattice data
  - compatible with multi-dimensional arrays

Lattice

- Lattice is a set of sample points in a space (geometry)
  - sample points represent a function over a domain
- Sample points may be organized in a grid (topology)
  - structured or unstructured
  - often represents spatial/temporal proximity
  - partitions topological space into cells
- Approximating function produces data values everywhere in the topological space
- Error function estimates data error everywhere

Lattice Example

- Topology defines a 2D mesh
- Sample points defined at mesh corners
- Geometry is 3D



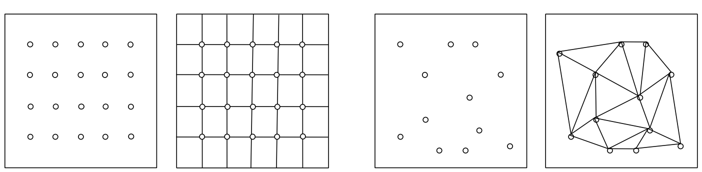
Topology

- Bridge between geometric position and physical storage location
- Defines adjacency relationships
- Basis for *cell view* of data
- Common topologies
  - rectilinear grids
  - curvilinear grids
  - unstructured grids

Topology/Geometry Examples

Regular sampling pattern

Unstructured pattern



No topology

Rectilinear topology

No topology

Tri-linear mesh topology

Processing by Geometry & Topology

- Geometry-based search
  - sample points in a *neighborhood* of some point in the domain
  - sample points within a region of the domain
- Topology-based search
  - points within a topological cell
  - points reachable via k edges in topology graph
- Topology can also be used to speed up geometric searches

Data Source

- Models the mapping of the *conceptual lattice to the physical data layer*
- *Models the notion of a computational space*
- *Based on viewing the physical layer as a multi-dimensional array*
- *PhysicalDataSource is a direct representation of a data file (or url)*

Data Source Framework

- *Data sources do not actually contain data; they just describe a conceptual view and how to get the data associated with that view*
- *PhysicalDataSource maps directly to a data file*
  - *defines dimensionality, sizes, record formats, etc.*
- *AttributeJoinDataSource combines 2 or more spatially symmetric data sources with different attributes*
- *BlockedDataSource spatially combines 2 or more data sources with matching attributes*

Implementation Overview

- Java is implementation platform
  - portability, distributed computing support, future potential
- Major Components
  - Data Source
  - Lattice
- Implementation guidelines critical for adequate performance

Implementation Guidelines

- Very large data sets impose difficult constraints
- Data access cost dominates, need to minimize
  - I/O costs to read the data
    - Avoid reading data that isn't actually needed
  - memory costs to store the data
  - cpu costs to access the data
    - Use lazy evaluation for data access
    - Avoid object creation related to individual data values
    - Minimize data copying
- Caching and pre-fetching important

Major Data Source Classes

- Three major families of classes
  - DataSource
    - definition of a *conceptual data store*
    - *does not actually contain data, defines how to get it*
  - Datum
    - collection of data values located at a point
    - mostly a *conceptual object; avoid actually creating Datums*
  - DataBlock
    - collection of data values for many adjacent points

Current Implementation Status II

- *Persistent DataSource definition*
  - *mySQL database version implemented*
  - *XML-based file definitions implemented*
- *Persistent Lattice definition not complete*
- *Performance testing harness implemented*

Conclusions

- We defined a Scientific Data Model that supports
  - *dimensional data (spatial and temporal)*
  - *multivariate and multidimensional data*
  - *implicit spatial and temporal relations*
  - *multisource and distributed data*
  - *multiresolution and adaptive resolution data*
- *We have a prototype system that indicates that*
  - *the model is implementable, and*
  - *may be efficient enough (within Java performance constraints)*

Future Work

- Our implementation does not yet fully support
  - multiresolution and adaptive resolution data
  - transparent distribution of data
  - distribution of processing
  - lots more