Tutorial - AM2

Large-Scale Array Analytics: Taming the Data Tsunami

Peter Baumann

Crowne Plaza Hotel
Glasgow, Scotland
24-28 October 2011

www.cikm2011.org
Large-Scale Array Analytics: Taming the Data Tsunami
CIKM 2011, Glasgow
Peter Baumann
Jacobs University

- international, multi-cultural
- 110 nations, English official language on campus
Array Research @ Jacobs U

- Jacobs University: international, multi-cultural
  - 110 nations, English official language on campus

- Large-Scale Scientific Information Systems research group
  - focus: large-scale n-D raster services & beyond
  - See www.jacobs-university.de/lsis

- Results
  - rasdaman raster („array“) DBMS
  - OGC standardization
    - Chair, coverage working groups
    - editor of 8+ stds, several candidate stds

OGC standardization
- Chair, coverage working groups
- editor of 8+ stds, several candidate stds
Roadmap

- Introduction
- Conceptual modelling
- Architecture
- Related Work
- Applications
- Wrap-up

Facing the Data Tsunami

“sensor” feeds

coverage server

[OGC-SWE]
Taming the Data Tsunami

Array-Intensive Methods: Differentiation

- multimedia databases
  - Analyse images, then drop them and work on auxiliary structure (i.e., feature vector)

- image processing
  - Advanced processing of rasters, but on main memory size objects

- image understanding, computer vision
  - Aiming at feature extraction etc. specific task
  - Again, not significantly beyond main memory sizes

- visual analytics
  - Visual display/interaction of analysis results
  - Again, main memory size limits
• Application specific. Parts represent an "image" in app context
• Abstract description; no explicit geometry and features
• Geometric description, attributes, features, viewing parameters
• Space and color discretisation
• Images als analog signals
• Optical signals as visual stimuli

Why Array Databases?

• "classical" database benefits for raster data:
  • data integration
  • flexibility
  • scalability
  • ...plus all further assets, like off-the-shelf tool support

• Unfortunately database people are soooo conservative
  • "images are matrices [...] which are stored as byte strings, ie, BLOBs"
  • Array databases fill this gap
Array Analytics

- **Array Analytics**: Efficient analysis on **multi-dimensional arrays of a size several orders of magnitude above evaluation engine’s main memory**
  - Typically in client/server setup
  - „Big Science“ on „Big Data“, both ad-hoc and long-tail
  - For this talk: „array“ = „raster“

- **Issues**:
  - Concepts: modeling, access interfaces (query languages)
  - Architecture: storage, processing, optimization
  - Scalability, usability, applications, standards

- *...obviously a typical database task (why didn’t we realize this earlier?)*

Who Needs Array Databases?

- **Sensor, image, statistics data**
  - **Life Science**: Pharma/chem, healthcare / bio research, bio statistics, genetics
  - **Geo**: Geodesy, geology, hydrology, oceanography, meteorology, earth system research, ...
  - **Engineering & research**: Simulation & experimental data in automotive/shipbuilding/ aerospace industry, turbines, process industry, astronomy, experimental physics, high energy physics, ...
  - **Management/Controlling**: Decision Support, OLAP, Data Warehousing, census, statistics in industry and public administration, ...
  - **Multimedia**: e-learning, distance learning, prepress, ...
Array Models: History

- Image partitioning, API access library [Tamura 1980]
- Fixed set of imaging operators [Chang, Fu 1980; Stucky, Menzi 1989; Neumann et al 1992]
  - scaling, rotation, edge extraction, thresholding, ...
- PICDMS [Chock, Cardenas 1984]
  - stack of images (identical resolution); operations corresponding to rasql "induced" ops; no nesting; no architecture
- ESRI, Oracle; Google, Microsoft, ...: ad-hoc solutions
Goal: enabling databases with support for massive n-D Sensor, Image, & Statistics Data [Baumann 1992+]

Starting point was user study:

\[ \text{how do imaging people model n-D array operations?} \]

- Most inspired by AFATL Image Algebra [Ritter et al 1990]

Algebra basis for conceptual model, storage mapping, & optimization

- Simplified: only arrays; reduced set of "pixel" types (atomic & nested records)
- Database-adjusted: small, closed set of primitives, safe in evaluation

Array Algebra Overview

- array = function: \( a: X \to F \) 
  (X n-D integer interval) 
  \( a = \{ (x,a(x)) : x \in X, a(x) \in F \} \)

- Core operations:
  - array constructor -- build array & initialize from cell expression
  - Condenser -- summarize over array, delivering a scalar 
    (using some commutative & associative summarization op)
  - Sorter -- slice array along a dimension, sort slices

- All else just shorthands: image addition, overlaying, statistics, ...
Array Operations: MARRAY

- **Array constructor**: \( \text{MARRAY}(\ e|_x, \ X, \ x) := \{ (x,f): f = e|_x, \ x \in X \} \)
  - for expression \( e|_x \)
    - potentially containing occurrences of \( x \), of result type \( F \)

- Example: image addition
  - \( a + b := \text{MARRAY}(a[x] + b[x], X, x) := \{ (x,f): f = a[x] + b[x], x \in X \} \)

- \( \rightarrow \) shorthands:
  - unary and binary "induced" operations
  - "whenever I have a pixel operation, I automatically have the corresponding image operation"
  - Image addition, comparison, component access, ...
    - \( a + b, \ a > b, \ a.\text{green}, ... \)

Array Operations: COND

- **Condenser**: \( \text{COND}(\ e|_{a,x}, \ o, \ X, \ x) := e|_{a,p_1} \circ e|_{a,p_2} \circ \ldots \circ e|_{a,p_n} \)
  - \( x \) visits each coordinate in \( X = \{ p_1, \ldots, p_n \} \)
  - \( e \) expression potentially containing \( a \) and \( p_i \)
  - \( \circ \) commutative, associative

- Example: "Sum over all cell values"
  - \( \text{add}(a) = \text{COND}(a[x], +, \text{sdom}(a), x) \)
    - \( = a[p_1] + a[p_2] + \ldots + a[p_n] \)
From Algebra To Query Language

- Data model: (multi-) sets ("collections") of typed arrays
- Data definition language rasdl [ODMG ODL]
  - Parametrised array constructor
- Retrieval and manipulation language rasql [ISO SQL92]
  - Set oriented, multidimensional operators
- Architecture streamlined towards piecewise processing of large objects
  - Tile streaming

<table>
<thead>
<tr>
<th>my_coll</th>
<th>OID</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>old 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>old 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>old 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>old 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>old 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

แรทเธอร์ ทร不起 ผล คือ

- typedef marray < unsigned char, [ 1:1024, 1:768 ] > XGA_Grey_Image;
- typedef marray < struct { unsigned char red, green, blue; }, [ *:*, *:* ] > RGB_Image;
- typedef marray < unsigned short, [ 1:1654, 1:* ] > G3_Fax;
- typedef marray < struct { double vx, vy; }, [ 0:*, 0:127, 0:63, 0:16 ] > ECHAM_T42_Windspeed;

Raster Type Definition

All C/C++ types, except pointers
The rasql Query Language

- **selection & section**
  
  ```sql
  select c[ :*, 100:200, :*, 42 ]
  from ClimateSimulations as c
  ```

- **result processing**
  
  ```sql
  select img * (img.green > 130)
  from LandsatArchive as img
  ```

- **search & aggregation**
  
  ```sql
  select mri
  from MRI as img, masks as am
  where some_cells( mri > 250 and m )
  ```

- **data format conversion**
  
  ```sql
  select png( c[ :*, :*, 100, 42 ] )
  from ClimateSimulations as c
  ```

---

Application Example: Histogram

- Histogram of an n-D array over 8-bit unsigned integer:
  
  ```sql
  select marray n in [0:255]
  values count_cells( image = n )
  from image
  ```

- changes cell type, dimension, domain
Storage Mapping

- **Task:**
  - materialise finite interval \( X \subseteq \mathbb{Z}^n \), find suitable (disk) access structure
  - Core structural property: Euclidean neighbourhood in \( \mathbb{Z}^n \)
  - Secondary, contents/app based: data density („sparsity“), data pattern, access pattern

- **Excursion: arrays in main memory**
  - Ex: APL [Iverson 1968]
  - **Assumption 1:**
    - access times independent from array position
    - \( \text{cost} \left( \left[ \text{a} \left[ \text{x} \right] \right] \right) = \text{const for all } \text{x} \)
  - **Assumption 2:**
    - access times independent from access sequence
    - \( \text{cost} \left( \left[ \text{a} \left[ \text{x} \right] ; \text{a} \left[ \text{y} \right] \right] \right) = 2 \times \text{const} \left( \left[ \text{a} \left[ \text{x} \right] \right] \right) = \text{const for all } \text{x}, \text{y} \)
Storage Mapping: Variants

- Coordinate-free sequence
  - BLOB (binary large object)
  - Costs mainly position/dimension dependent

- Sequence independent, coordinates explicit
  - ROLAP
  - Costs not position correlated, but high

- Imaging, multidimensional OLAP
  - Partitioning, sequence within partition
  - Costs low for bulk access, usually not location correlated

Tiled Array Storage

- partition multidimensional object
  → multidimensional tiles
  - Tile = subarray
    [Widmann 2001, Furtado 2002]
  - Regular tiling = mosaicking [imaging, geo], chunking [Sarawagi, DeWitt, Stonebraker]

- Tiles form unit of access in persistent store
  - Ex: BLOB in relational database
  - Compression, geo index
Benchmarks: Tiling Strategy

Operand: 3-D MDD object
Operation: Z cut
selectivity: 1.6 %

tomo_sliced 153x256
time:

tomo_cubed 32x32x32
time:

Comparison: BLOB Read Performance

- Optimal tuning per system
- OS competitors often better!

MySQL ARCHIVE
- PostgreSQL, b=8k, l=2k
- SystemA, CHUNK=8k
- SystemB, p=16k

PostgreSQL, b=8k, l=2k
SystemA, CHUNK=32k
SystemB, p=16k
Comparison: Time to Read (Deduced)

- performance varies by two orders of magnitude!
- @100K / MySQL vs @10K / SystemB

![Graph showing time to read performance variations](image)

Tiling Strategies

- Goal: faster tile loading by adapting storage units to access pattern
- Tiling classification [Furtado+ 1999] based on degree of alignment

![Tiling strategies diagram](image)
Tiling Strategies

- **Goal:** faster tile loading by adapting storage units to access pattern
- **Tiling classification** [Furtado+ 1999] based on degree of alignment
- **Issues**
  - When is tiling optimal? Tiling strategies?
- **3 sample tiling strategies** [Furtado 1999]:

Storage Layout Language

- **Goal:** Support ad-hoc storage tuning
- **Approach:** array storage layout sub-language extending `insert` statement [Baumann+ 2010]
- **Ex:**
  ```sql
  insert into MyCollection
  values ...
  tiling area of interest [0:20,0:40], [45:80,80:85]
  tile size 1000000
  index d_index
  storage array
  compression zlib
  ```
Adding Tertiary Storage

- tape archives for near-line access [Sarawagi, Stonebraker 1994]

- Problem: respect spatial clustering
  - Access locality (long positioning times!)

- Approach: super tiles = all tiles of particular index node [Reiner 2001]
  - Natural unit, comfortable to handle

Coffee Break!
Architecture II: Query Processing

Architecture

- Client Communication Layer
- Server Communication Layer
- QL Parser
- Optimizer
- Executor
- Index
- Cache & TA
- Catalog
- Base DBMS Interface

Conventional base DBMS

- alphanumeric data

Raslib / rasj rasql

P. Baumann :: Large-Scale Array Analytics :: CIKM 2011
Query Processing: Overview

- Parsing
- Normalisation
- Optimization
  - Common subexpression elimination
- [Generate query plan]
- Tile-based evaluation

```sql
select a < avg_cells( b + c )
from a, b, c
```

Benchmarks: Data Access

[Ritsch 2000, Widmann 2001]
**Can't We Do That Object-Relationally?**

- **Marray** is not a type, but a **type constructor**

- **Cf. Stack:**
  - `Stack<>` is type constructor
  - `Stack<int>, stack<float>, ...` are concrete, instantiated types

- Relational model does not know type constructors → hard to integrate
  - does not even know user-defined attribute types

- Object-relational extensions allow user-defined data types, however **not** type constructors → no benefit

- Actually, whole engine stack needs reimplementation
  - Sub-page tuples vs multi-page (multi-disk!) arrays
Query Rewriting

```
select avg_cells( a + b )
from a, b
```

```
select avg_cells( a )
+ avg_cells( b )
from a, b
```

- understood: heuristic optimization
  - 150 rules in rasdaman [Ritsch 2002]
- partially understood: cost-based optimization

Just-In-Time Compilation

- Observation: interpreted mode slows down
- Approach:
  - cluster suitable operations
  - compile & dynamically bind
- Benefit:
  - Speed up complex, repeated operations
- Variation:
  - compile code for GPU

```
for x in (float_matrix)
return x*x*...*x
```
GPU Processing

- Observation: pixelwise operations costly

- Approach (patented):
  - cluster suitable operations
  - Generate GPU code
  - Spawn GPU process

- Advantages:
  - keep CPU + GPU humming
  - # GPU cores >> # CPU cores
  - GPU driver schedules

- Preliminary observation:
  performance independent from #ops for up to ~100 ops

GPU Processing

[Stancu-Mara 2008]

Optimisation Does Pay Off!

- Complex queries give more space to optimizer

- Example 1: Typical OGC Web Map Service query:

```sql
select jpeg(
    scale(bild0[...],[1:300,1:300]) * {1c, 1c, 1c}
    overlay ((scale(bild1[...],[1:300,1:300])<71.0)) * {51c, 153c, 255c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 2) * {1c, 1c, 1c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 5) * {1c, 1c, 1c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 7) * {102c, 102c, 102c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 3) * {191c, 242c, 128c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 4) * {191c, 255c, 255c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 1) * {0c, 255c, 255c}
    overlay bit(scale(bild2[...],[1:300,1:300]), 0) * {102c, 102c, 102c}
)
from ...
```
Optimization Does Pay Off!

- Example 2: real-time WMS zoom/pan/styling
  - 1 background, 1 bathymetry, 3*RGB
  - www.earthlook.org

Optimization

- Adaptive tiling
- Adaptive compression
- Multi-dimensional indexing
- Distributed query processing
- Query rewriting
- Pre-aggregation
- Physical operator clustering
- Transparent tape integration
- Just-in-time compilation
- GPU processing
- Tile caching
- ...
Query Parallelisation

- **easy: inter-query parallelization** (one client – one server process)
  - Long-runners don't block service
  - higher throughput

- **Non-trivial: intra-query parallelization** (one client – several server processes) [Hahn 2003]
  - Idea: tiles dynamically assigned to processors
  - Non-trivial array index patterns?
Geo Service Standardization

- OGC (Open GeoSpatial Consortium) driving geo service standards
  - Web-based modular, open, interoperable geo services
  - Liaisons with ISO TC 211, OASIS, CGI/IUGS; ...
  - Consensus body, specs tested before released (eg, testbeds)
  - www.opengeospatial.org

- Array data special category of coverage in OGC / GIS speak
  - Web Coverage Service Standards Working Group (WCS.SWG)
  - Web Coverage Processing Service Group (WCPS)
  - Coverages WG
  - Metocean Domain Working Group
  - GALEON (Geo-interface to Atmosphere, Land, Earth, Ocean, NetCDF) OGCnetwork

(Part of) The OGC Quilt
What Is a Coverage, After All?

- Coverage =
  n-D "space/time-varying phenomenon"
  
  [ISO 19123, OGC AT6]

WCS Core Functionality

- In Core, simple data access (more in extension packages):

  subset = trim | slice
Use Case: Satellite Image Time Series

[Diedrich et al 2001]

WCS Suite: The Big Picture
Web Coverage Processing Service

- OGC WCPS standard, adopted 2008 [OGC 08-068r2]
  = aka “XQuery for multi-dimensional coverages”
  - image & signal processing, statistics
- (semi) formal algebraic semantics
- Safe in evaluation
- Expression nesting → unlimited complexity

WCPS By Example

- "From MODIS scenes $\mathbf{M}_1$, $\mathbf{M}_2$, and $\mathbf{M}_3$, the absolute of the difference between $\text{red}$ and $\text{nir}$, in HDF-EOS"

\[
\text{for } c \text{ in } ( \mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3 ) \text{ return encode(}
\text{abs( } c.\text{red} - c.\text{nir} ,
\text{"hdf" )}
\]
WCPS By Example

- "From MODIS scenes M1, M2, and M3, the absolute of the difference between red and nir, in HDF-EOS"
  - ... but only those where nir exceeds 127 somewhere

```python
for $c$ in ( M1, M2, M3 )
where
    some( $c.nir > 127$ )
return
    encode
        abs( $c.red - c.nir$ ),
        "hdf"
```

WCPS By Example

- "From MODIS scenes M1, M2, and M3, the absolute of the difference between red and nir, in HDF-EOS"
  - ... but only those where nir exceeds 127 somewhere
  - ... inside region R

```python
for $c$ in ( M1, M2, M3 ),
    $r$ in ( R )
where
    some( $c.nir > 127$ and $r$ )
return
    encode
        abs( $c.red - c.nir$ ),
        "hdf"
```
Sample WCPS-Based C/S Architecture

- ChartLink (Envitia Ltd, UK)
  - rich geo client
  - assembles WCPS queries, wrapping into visual functionality

- rasdaman (Jacobs U, rasdaman GmbH)
  - WCPS over various protocols
  - petascope for request translation and geo semantics resolution
  - rasdaman returns images (or scalars)

[ESA VAROS project, 2010]
Climate Modelling

- Example: ECHAM T42 (cf. video)
- 50+ physical parameters ("variables"): temperature, wind speed x/y, humidity, pressure, CO2, ...
- 2.5 TB per variable

<table>
<thead>
<tr>
<th>dimension</th>
<th>extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitude</td>
<td>128</td>
</tr>
<tr>
<td>Latitude</td>
<td>64</td>
</tr>
<tr>
<td>Elevation</td>
<td>17</td>
</tr>
<tr>
<td>time (24 min per time slice)</td>
<td>2,190,000 (200 years)</td>
</tr>
</tbody>
</table>

"Even with multi-terabyte local disk subsystems and multi-petabyte archives, I/O can become a bottleneck in HPC."
-- Jeanette Jenness, LLNL, ASCI-Project, 1998
EarthServer: Big Analytics on Big Data

- **Mission:** to enable standards-based ad-hoc analytics on the Web for Earth science data
  - scalable to Petabyte/Exabyte volumes
  - directly manipulate, analyze & remix any-size geospatial data
- **Core idea:** integrated query language for all spatio-temporal coverage data
- **Goal:** to establish OGC standards based client & server technology
- **Funded by EU FP7-INFRA**
  - Started Sep 1, runtime 3 years, 5.38m EUR budget, 11 partners

EarthServer Lighthouse Applications

- 100+ TB per site, accessible for direct analytics
- front-end to existing archives - no new archives

- **EO**
  - snow & land ice
  - x/y + x/y/t

- **Geology**
  - 3D geological models
  - x/y + x/y/z

- **Oceanography**
  - EO + marine model runs + in-situ
  - x/y + x/y/z + x/y/z/t

- **Meteorology**
  - climate variables
  - x/y/z/t/variables

- **Planetary Sci**
  - Mars geology
  - x/y + x/z + y/z
EarthServer: Main Innovations

- Integrated coverage, feature, and metadata queries, including all OGC coverage types
- Transparent queries over heterogeneous file archives and databases
- Paving the way for Petabyte services: cloud distribution, parallelization, supercomputers
- Comprehensive OGC standards support for coverage data and services

Vision: barrier-free „mix & match” access to multi-source, any-size geo data

Cosmological Simulation

- Modelling domain: 4D
  - Dark matter (highest mass factor in universe)
  - Baryonic matter (stars, gas, dust, …)
  - Coupled simulation: particle + fluid
- Results: 3D/4D cutouts from universe
  - Eg, 64 Mpc³ (1 pc = 3.27 light years)
- Screenshots: AstroMD [Gheller, Rossi 2001]
Cosmology (contd.)

- Guided retrieval:
  - Selection of objects and their attributes (cell components)
  - Interactive setting of trim operations per dimension
  - Augmented with induced operations
- Suitable for expert users
- Details: cosmolab.cineca.it

---

Human Brain Imaging

- Research goal: to understand structural-functional relations in human brain
- Experiments capture activity patterns (PET, fMRI)
  - Temperature, electrical, oxygen consumption, ...
  - Lots of computations → "activation maps"
- Example: "a parasagittal view of all scans containing critical Hippocampus activations, TIFF-coded."

```sql
select tiff( ht[ $1, *,*, *:* ] )
from HeadTomograms as ht,
     Hippocampus as mask
where count_cells( ht > $2 and mask )
     / count_cells( mask ) > $3
```

$1 = slicing position, $2 = intensity threshold value, $3 = confidence
Gene Expression Analysis

- Gene expression = reading out genes for reproduction
- Research goal: capture spatio-temporal expression patterns in Drosophila

```
select jpeg( scale( {1c,0c,0c}*e[0,*,*], 0.2 
+{0c,1c,0c}*e[1,*,*], 0.2 
+{0c,0c,1c}*e[2,*,*], 0.2 ) )
from EmbryoImages as e
where oid(e)=193537
```

Summary: Domains Investigated

- Geo
  - Environmental sensor data, 1-D
  - Satellite / seafloor maps, 2-D
  - Geophysics (3-D x/y/z)
  - Climate modelling (4-D, x/y/z/t)
- Life science
  - Gene expression simulation (3-D)
  - Human brain imaging (3-D / 4-D)
- Other
  - Computational Fluid Dynamics (3-D)
  - Astrophysics (4-D)
Related Work: Brief History

- Image partitioning, API access library [Tamura 1980]
- Fixed set of imaging operators [Chang, Fu 1980; Stucky, Menzi 1989; Neumann et al 1992]
  - scaling, rotation, edge extraction, thresholding, ...
- PICDMS [Chock, Cardenas 1984]
  - stack of images (identical resolution); operations corresponding to rasdaman "induced" ops; no nesting; no architecture
- rasdaman [Baumann+ 1991+]: algebra, QL, architecture
- „Call to order“ [Maier 1993]
- AQL, AML, MQL: conceptual models
- Sarawagi/Stonebraker: tertiary storage
- ESRI, Oracle; Google, Microsoft, ...
  - Mostly Geo (Remote Sensing), some Space, practically no Life Science motivation
- TerraLib, MonetDB, SciDB, ...

‡ see next
Related Work: Systems

- **Oracle GeoRaster**
  - 2D, no QL integration

- **PostGIS Raster**
  - Excellent QL integration
  - 2D, no tile management, no storage layout tuning, no adaptive tile streaming, no raster query optimization; utilizes small tiles; ... scalability?

- **MonetDB (column store DBMS)**
  - n-D arrays under development
  - Arrays as first-class citizens – array similar to table

- **SciDB**
  - n-D arrays announced, components demoed, under development
  - Mingles logical with physical aspects on QL level
Related Work: Tiling

- Partitioning common in imaging & geo data
  - Tiling, mosaicking, ...
  - e-Science often uses irregular partitioning
  - Array databases
    - Regular „chunks“ [Stonebraker, Sarawagi 1996], refined by [Rotem et al 2008]
    - Also regular: TerraLib [Vinas+ 2007], MonetDB [Ballegooij+ 2005], PostGIS Raster [Racine 2010], ESRI ArcSDE, Oracle 11g
    - SciDB [Cudre-Maroux et al 2009]: 2-level approach, regular chunking, redundancy
    - rasdaman [Baumann 1994, Furtado+ 1999]: arbitrary partitioning

Related Work: Applications

- MS SQL Server / SDSS SkyServer [Gray et al, ]
  - Recently: MonetDB / SDSS SkyServer [Ivanova et al, DBBD 2007]
  - Emphasis on point objects and proximity queries, no arrays in “top 20 queries”
The Big Picture

- Large-scale array services important + growing field
  - Currently driven by geo services
  - Largely neglected challenge to databases
  - largest single DB objects ever!
- Service providers & users demand it
  - "2D, 3D imagery next great challenge in geo databases" [Xavier Lopez, Oracle]
- Can translate most features from alphanumeric databases (and benefit):
  - Declarative, optimizable query language
  - formal semantics definition
  - Suitable storage architecture
- Many open issues, such as:
  - what expressive power? Primitives?
  - architecture
  - optimization
  - standardized benchmarks
Use Case: Reverse Lookup

"all clinical trials of drug X where patient temperature > 40° C within the first 48 hours."

Conclusion

- Array databases form nucleus for large-scale scientific data analytics
  - n-D arrays found in earth, space, life sciences, business, ...
  - Emerging "next wave" – cf XLDB, Array Databases workshop @ EDBT/ICDT (www.rasdaman.com/ArrayDatabases_Workshop)
    - Our research: flexible, scalable raster services & beyond
- DB technology can contribute significantly,
  - Flexibility, scalability, information integration, ...
- ...but must transcend traditional (table-driven) viewpoints
  - QL primitives, architectures, ...