Roadmap

- Introduction
- Installation
- Architecture
- Querying & Hands-on Session
- Wrap-up & discussion

Disclaimer:
This is a tutorial which has been presented at VLDB 2015. As such, it is supplementary material and not necessarily optimal for self-study. See www.rasdaman.org for ample ancillary material.
Introduction
BIG EARTH DATA
The Digitized Planet
Structural Variety in Big Data

- Stock trading: 1-D sequences (i.e., arrays)
- Social networks: large, homogeneous graphs
- Ontologies: small, heterogeneous graphs
- Climate modelling: 4D/5D arrays
- Satellite imagery: 2D/3D arrays (+irregularity)
- Genome: long string arrays
- Particle physics: sets of events
- Bio taxonomies: hierarchies (such as XML)
- Documents: key/value stores = sets of unique identifiers + whatever
- etc.
Array Analytics

- Array Analytics :=
  *Efficient analysis on multi-dimensional arrays of a size several orders of magnitude above evaluation engine‘s main memory*

- Essential data property: n-D Euclidean neighborhood
  - Secondary: #dimensions, density, ...

- Operations: Linear Algebra++

[EDBT/ICDT Array Databases Workshop, 2011]
A Simple Example

- Divergent access patterns for ingest and retrieval
- Server must mediate between access patterns
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 have been soooo conservative
  - "images are matrices [...] which are stored as byte strings, ie, BLOBs"
  - „this is NOT SQL!“
A Brief History of Array DBMSs

first appearance in literature (not first implementation)
Array Analytics Research @ Jacobs U

- Large-Scale Scientific Information Systems research group
  - Flexible, scalable n-D array services
  - www.jacobs-university.de/lsis

- Main results:
  - pioneer Array DBMS, rasdaman
  - standardization: OGC Big Geo Data, ISO SQL

Hiring PhD students, PostDocs
rasdaman: Agile Array Analytics

- „raster data manager“: n-D arrays in SQL
- Array Algebra [NGITS 1998]
- Declarative, optimizable QL
- Scalable, parallel architecture
  - „tile streaming“
Installation
Options for Today

- Install from source: [www.rasdaman.org](http://www.rasdaman.org)
  - Prerequisite: Linux laptop

- Install RPM (CentOS 6, 7)
  - Prerequisite: Linux laptop

- Boot from USB stick
  - Prerequisite: laptop

- Run demo queries from browser
  - Prerequisite: laptop
Coffee Break!
Query Language Intro
The Multidimensional Data Model
The rasdaman Data Model

- **Data model:** tables of typed n-D arrays
  - Array type knows cell type, extent/dimension

- **Original rasql:** Array + system attribute OID
  - ODMG speak: „collections“ = relations
  - Typed; any C/C++ type for cells
  - Ex: `typedef Marray< int, [0:255,0:*] > GreyImage;`

- **ISO SQL/MDA (see later):** tight DDL/DML integration with SQL
  - Ex: `create table LandsatScenes(
    id: integer not null, acquired: date,
    scene: row( band1: integer, ..., band7: integer ) array [ 0:4999,0:4999] )`

<table>
<thead>
<tr>
<th>MyColl</th>
<th>OID</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>oid 1</td>
<td><img src="image" alt="Red cube" /></td>
</tr>
<tr>
<td></td>
<td>oid 2</td>
<td><img src="image" alt="Yellow cube" /></td>
</tr>
<tr>
<td></td>
<td>oid 3</td>
<td><img src="image" alt="Yellow cube" /></td>
</tr>
<tr>
<td></td>
<td>oid 4</td>
<td><img src="image" alt="Green cube" /></td>
</tr>
<tr>
<td></td>
<td>oid 5</td>
<td><img src="image" alt="Green cube" /></td>
</tr>
</tbody>
</table>

metadata table:
- att 1
- att 2
- att n
- key 1
- oid 1
- key 2
- oid 2
- key 3
- oid 3

**will use rasql here**
QL in a Nutshell

- Remember: tables with single, unnamed column of arrays

- trimming & slicing

```sql
select a[*:*,:100:200,:10] from AvgLandTemp as a
```

- result processing

```sql
select img * (img.green > 130) from NIR as img
```

- search & aggregation

```sql
select mr from MRScan as mr, masks as m where some_cells(mr > 250 and m)
```

- data format conversion

```sql
select encode(a[*:*,:*,:*,10], "png") from AvgLandTemp as a
```
QL Foundation: Array Algebra

- Starting point: domain studies
  - ISO Computer Graphics Reference Model, Visualization Reference Model, image processing languages, AFATL Image Algebra, etc.

- Result: minimal algebra for model, QL, storage mapping, optimization
  - Array iteration implicit → no explicit loops → declarative, safe

- QL = SQL with array expression [VLDBJ1994, NGITS 1998]:
  - Array constructor -- build array & initialize from cell expression
  - Array condenser -- summarize over array, delivering a scalar
  - Array sorter – reorder array slices
  - ...all else can be reduced to these
Array Algebra Ops: Constructor

- Define a new array (with extent), initialize cells

```
marray x in [0:99], y in [0:99]  
values x+y
```

- Shorthands:
  - Subsetting (trim, slice):
    ```
a[ x0:x1, y0:y1, t ]
```
  - „induced“ operations: for every cell operation, offer same on arrays
    ```
a.red + 5
```
    - cell component access
    - arithmetic, boolean, exponential, trigonometric ops,...
Array Algebra Ops: Condenser

- **Summarize over (part of) an array**

  \[
  \text{condense} + \\
  \text{over } x \ in \ [0:99], \ y \ in \ [0:99] \\
  [ \text{where } P ] \\
  \text{using } a[x,y]
  \]

- **Shorthands:**
  - usual suspects: \textit{count, sum, avg, max, min, some, all}

  \[
  \text{max\_cells}( a )
  \]
Sample Combinations

- **Matrix multiplication (here for simplicity: self-join)**

```plaintext
select marray i in [0:m], k in [0:p]
values condense +
over j in [0:n]
using a [ i, j ] * b [ j, k ]
from lena as a, lena as b
```

- **Histogram**

```plaintext
select marray bucket in [0:255]
values count_cells( lena = bucket )
from lena
```
Hands On Querying
Let's Get Hands On!

- **Online:**
  - For images: prefix query with `image>>`
  - For diagrams: prefix query with `diagram>>`
  - Otherwise (text output): no prefix

- **USB sticks:**
  - `start_rasdaman.sh` → open browser, see same page as above, continue as above

- **Compiled from source: use command line:**
  - `start_rasdaman.sh`
  - For images: `$ rasql -q "select ... " --out image`
  - Otherwise: `$ rasql -q "select ... " --out string`
What Data is Available?

- Collections hold array objects
- Get a list of available collections:

  ```sql
  select c
  from RAS_COLLECTIONNAMES as c
  ```

  - Virtual collection of 1-D char arrays

- In this tutorial:
  - Mostly using 3D global land temperature timeseries
  - Most collections contain 1 row
Array Schema & Other Information

- Get a list of available collections:
  
  ```
  select c
  from RAS_COLLECTIONNAMES as c
  ```

- Virtual collection of 1-D char arrays
  
  ```
  select oid(c)
  from AvgLandTemp as c
  ```

- `oid()` – array identifiers
  
  ```
  select sdom(c)
  from AvgLandTemp as c
  ```

- `sdom()` – list of axis boundary intervals (ie, integer pairs)
  
  ```
  select sdom(c)
  from AvgLandTemp as c
  ```

- `dbinfo()` – physical storage information
Trimming & Slicing

- Array subsetting:
  - Trim takes interval, retains dimension of result
  - Slice takes point, each slicing reduces dimension of result

- Ex (substitute numbers for x, y, t):
  - Single point (0D = scalar)
    
    ```
    select c[ x, y, t ]
    from AvgLandTemp as c
    ```

  - 1D timeseries
    
    ```
    select c[ x, y, t1:t2 ]
    from AvgLandTemp as c
    ```

  - 2D timeslice
    
    ```
    select c[ *:*:*:*:*, t ]
    from AvgLandTemp as c
    ```
Induced Operations

- = overloaded array cell operators, as aka shorthands

- Ex: „Convert all values from Celsius to Kelvin for one year at position x/y“

\[
\text{select encode( } \text{c[x,y,0:11]} + 273.15, \ "csv" \ ) \\
\text{from AvgLandTemp as c}
\]

- Ex: Pixel-wise „band math“ in remote sensing

\[
\text{select encode( c.0 - c.1, \ "png" \ )} \\
\text{from NIR as c}
\]

- Ex: Real-life example: NDVI (vegetation index) from false-color image

\[
\text{select encode( } \text{(char) } (((\text{float)c.0 - c.1}) / ((\text{float)c.0 + c.1})) > 0.7) \times 255, \ "png") \\
\text{from NIR as c}
\]
Induced Operations /contd.

- Conditional evaluation: SQL case statement, extended to arrays
  - Ex: „color code output based on cell values, null values as black (ie, transparent)“

```sql
select case
    when c[1600:2200,150:550,7] = 99999 then {255c,255c,255c}
    when c[1600:2200,150:550,7] < 18 then {0c,0c,255c}
    when c[1600:2200,150:550,7] < 24 then {0c,255c,0c}
    else {255c,0c,0c} end
from AvgLandTemp as c
```

- In summary: all unary, binary, n-ary cell operations can be induced
  - record access, arithmetic, logarithmic, trigonometric, comparison, Boolean, cast, case
Array Construction /contd.

- Subsetting & induced ops expressible through basic array constructor, as per algebra:

- Ex: „2D 100x100 cutout in space, slice at time t=0“

```sql
select encode(
    marray x in [0:99], y in [0:99]
    values c[x,y,0],
    "png" )
from AvgLandTemp as c
```

- Ex: „1D array of bi-monthly average temperatures at a certain location“

```sql
select encode(
    marray t in [0:5]
    values ( c[1888, 369, 2*t]
    + c[1888, 369, 2*t+1] ) / 2,
    "png" )
from AvgLandTemp as c
```
Aggregation

- First, shortcuts: `min_cells`, `max_cells`, `avg_cells`, `all_cells`, ...
  - Ex: „Minimum temperature of all months in first year, for location (100,200)“

```sql
select min_cells( c[ 100, 200, 0:11 ] )
from AvgLandTemp as c
```

- Next, basic condenser as per algebra
  - Ex: „Count number of months when average temperature over a particular area exceeds threshold“

```sql
select condense +
over t in [0:4]
where avg_cells( c[1800:1900, 300:400, t ] ) > 15
using 1
from AvgLandTemp as c
```
User-Defined Functions (UDFs)

- external code dynamically linked into server, callable from query
  - Leverage existing libraries within rasql queries difficult to represent as queries
- rasdaman: UDF API = client API + auto-generated adapter → easy to use
  - integrated with server-side tile management, parallelization, ...
- Demo only available on self-installed version, not over Web; before running query compile UDF code:
  1) Open terminal (Applications → Utilities → Terminal)
  2) $ cd ~/rasdaman/share/rasdaman/udf
  3) $ make && stop_rasdaman.sh && start_rasdaman.sh
- Ex: use OpenCV for histogram equalization on RGB image

```sql
select encode(
    cv.equalize_hist( c[0:999,0:999]/16 ), "png" ),
from AGDC_2D as c
```

1) Open terminal (Applications → Utilities → Terminal)
Cloud Demo

- Open URL shown
- 1 TB of Earth science timeseries data
- Run parallel queries in Amazon cloud
Architecture
Query Processing

- Clear separation: set vs array trees
  - Arrays as 2nd order attributes
- Extensive optimization
- Tile-based evaluation

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

- Goal: faster tile loading by adapting storage units to access patterns
- Approach: partition n-D array into n-D partitions ("tiles")
- Tiling classification based on degree of alignment [ICDE 1999]

chunking [Sarawagi, Stonebraker, DeWitt, ...]
Why Irregular Tiling?

- e-Science often uses irregular partitioning

[OpenStreetMap]

[Centrella et al: scidacreviews.org]
Adaptive Tiling

- Sample tiling strategies [ICDE 1999]:
  - regular
  - directional
  - area of interest

- rasdaman storage layout language [SSTDM 2010]

```plaintext
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
```
Architecture

Web clients (m2m, browser)

Internet

rasdamann

geo services

rasserver

File system
database

external files

distributed query processing
No single point of failure

alternative storage

[SSTD 2013]
Parallel / Distributed Query Processing

```sql
select
    max((A.nir - A.red) / (A.nir + A.red))
- max((B.nir - B.red) / (B.nir + B.red))
- max((C.nir - C.red) / (C.nir + C.red))
- max((D.nir - D.red) / (D.nir + D.red))
from A, B, C, D
```

1 query → 1,000+ cloud nodes

[SIGMOD DANAC 2014]
Parallel / Distributed Query Processing

- Online demo at http://earthlook.flanche.net/vldb_cloud_demo
  - ~1TB of 3D Landsat 5 time series data (2003-2011) over ACT area
  - 6 bands: blue, green, red, nir, mir1, mir2 (in rasql 0, 1, 2, ...)
  - distributed over 9 Amazon nodes
Wrap-Up & Discussion
EarthServer: Datacubes at Your Fingertips

- Operational Agile Analytics on 1+ Petabyte space/time datacubes
  - Earth Science (3D sat image timeseries, 4D weather); Planetary Science

- Based on & extending rasdaman
  - integrated data/metadata search
  - performance enhancements

- Intercontinental initiative: EU+US+AUS

- www.earthserver.eu
Science & GIS Tool Interfacing

- General-purpose scientist tools:
  - Python, R, Java, C++

- Geo tools:
  - MapServer, GDAL, QGIS, OpenLayers, NASA WorldWind, ...

- Open Geospatial Consortium (OGC) Web Coverage Service (WCS) Core Reference Implementation
  - OGC‘s „Big Geo Data“ standard
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)
  - Statistics (n-D)
Standardization: Geo Raster QL

- OGC Web Coverage Processing Service (WCPS)

- high-level geo raster query language; adopted 2008
  - Integration with XQuery
  - Geo semantics → variety of grid types:

Standardization: SQL

[SSDBM 2014]

Information technology — Database languages — SQL — Part 15: Multi-Dimensional Arrays (SQL/MDA)

create table LandsatScenes(
    id: integer not null, acquired: date,
    scene: row( band1: integer, ..., band7: integer ) mdarray [ 0:4999,0:4999 ] )

select id, encode(scene.band1-scene.band2)/(scene.nband1+scene.band2), „image/tiff“ ) from LandsatScenes
where acquired between „1990-06-01“ and „1990-06-30“ and
    avg( scene.band3-scene.band4)/(scene.band3+scene.band4)) > 0
„one size does not fit all“

...holds for the above sentence, too.

We need to inspect every case individually, and arrays fit nicely into SQL world.

NB: Geo world desperately tries to get away from silos, striving for (logical) data integration
Conclusions

- **Array Analytics**: support for a core category of „Big Data“ in sci & eng
  - Sensor, image, simulation, statistics data
  - Signal/image processing, statistics, Linear Algebra

- **DBMSs contribute flexibility, scalability, information integration, ...**

- rasdaman: pioneer ADBMS, in industrial use
  - PB of operational databases, 1,000+ nodes
  - OpenHub: rasdaman community @ 10m US$ value

- **See us:**
  - [www.rasdaman.org](http://www.rasdaman.org), [www.jacobs-university.de/lsis](http://www.jacobs-university.de/lsis), [www.earthserver.eu](http://www.earthserver.eu)