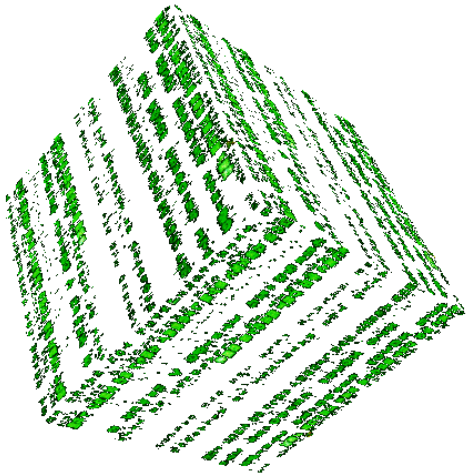


# Python
# Winding Itself Around Datacubes

FOSDEM 2017, Brussels

**Siddhart Shukla, Vlad Merticariu, Peter Baumann**

Jacobs University | rasdaman GmbH

baumann@rasdaman.com

[gamingfeeds.com]

[co-funded through EU EarthServer-2]

# BIG EARTH DATA

## The Digitized Planet

# Jacobs MSc in Data Engineering

Machine Learning – Big Data – Cloud Computing – Visualization – and more...

all-English, international campus – research involvement – strong industry connections

# Roadmap

- Array Databases
- Python 1: OWSLIB
- Python 2: general array database backend
- Summary

# Array Databases

# 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.

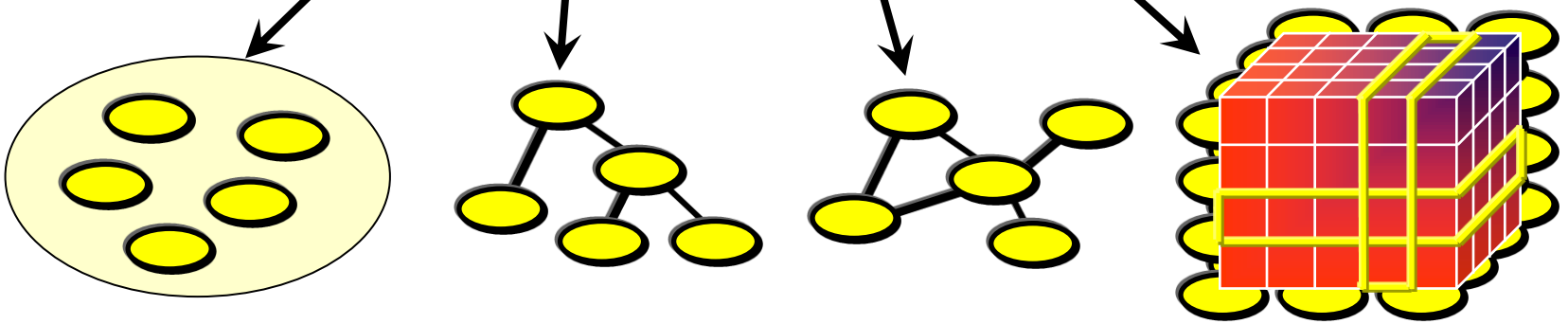
# Structural Variety in Big Data

- arrays
- graphs
- graphs
- arrays
- arrays
- arrays
- sets
- hierarchies
- sets
-

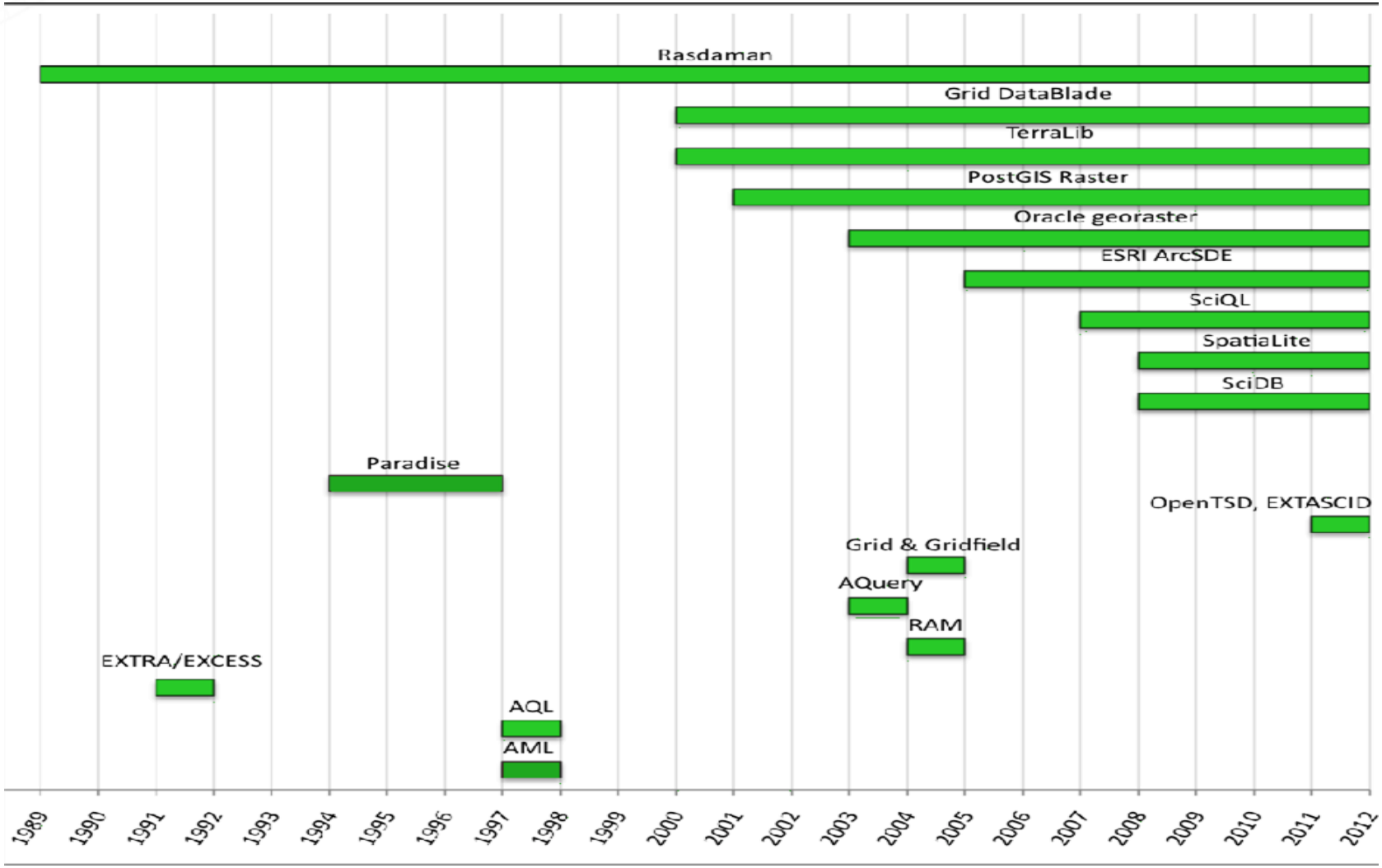
# Structural Variety in Big Data

sensor, image [timeseries], simulation, statistics data

**sets + hierarchies + graphs + arrays**



# A Brief History of Array Databases



# Array SQL

ISO/IEC JTC 1/SC 32

Date: 2014-06-04

WD 9075-15:2014(E)

**Information technology — Database languages — SQL —**

**Part 15:**
**Multi-Dimensional Arrays (SQL/MDA)**

ISO/IEC JTC 1/SC 32/WG 3

The United States of America (ANSI)

*Technologies de l'information — Langages de base de données — SQL —*

*Partie 15: Tableaux multi-dimensionnels (SQL/MDA)*

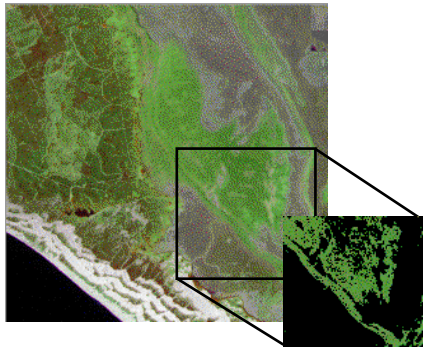
```
create table LandsatScenes(
    id: integer not null, acquired: date,
    scene: row( band1: integer, ..., band7: integer ) mdarray [ 0:4999,0:4999] )
```

```
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
          mdavg( scene.band3-scene.band4)/(scene.band3+scene.band4)) > 0
```

# rasdaman

= „raster data manager“: SQL+ n-D arrays

- pioneer Array Database System
- Scalable parallel “tile streaming” architecture
- www.rasdaman.org



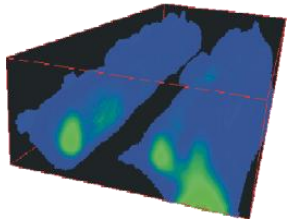
- Mature, in operational use
  - Ex: www.planetserver.eu
  - OSGeo Live

- OGC WCPS, ISO SQL/MDA blueprint

# rasQL in a Nutshell

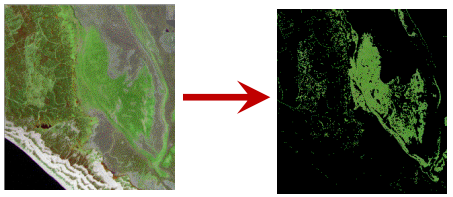
- trimming & slicing

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



- result processing

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

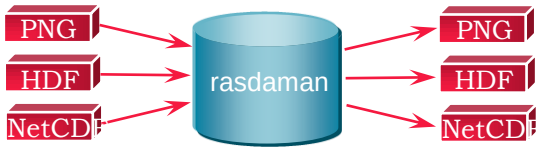


- search & aggregation

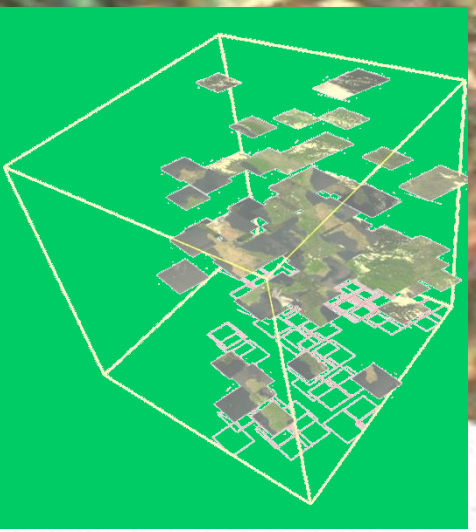
```
select mr
from   MRScan as mr, mask as m
where  some_cells( mr > 250 and m )
```

- data format conversion

```
select encode( a[*:*,*:*,10], "png" )
from AvgLandTemp as a
```



# Direct Data Visualization



```
select
  encode(
    struct {
      red:   (char) s.b7[x0:x1,x0:x1],
      green: (char) s.b5[x0:x1,x0:x1],
      blue:  (char) s.b0[x0:x1,x0:x1],
      alpha: (char) scale(d.elev,20)
    },
    "image/png"
  )
from SatImage as s, DEM as d
```

[JacobsU, Fraunhofer; data courtesy BGS, ESA]

# Linear Algebra Ops

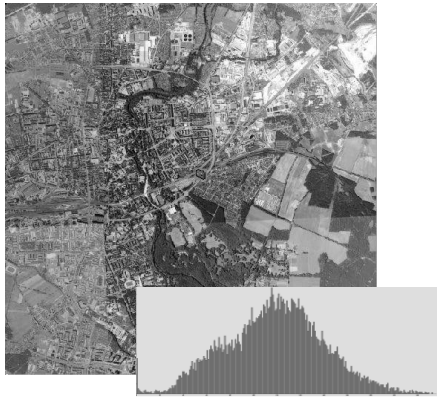
- Matrix multiplication

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

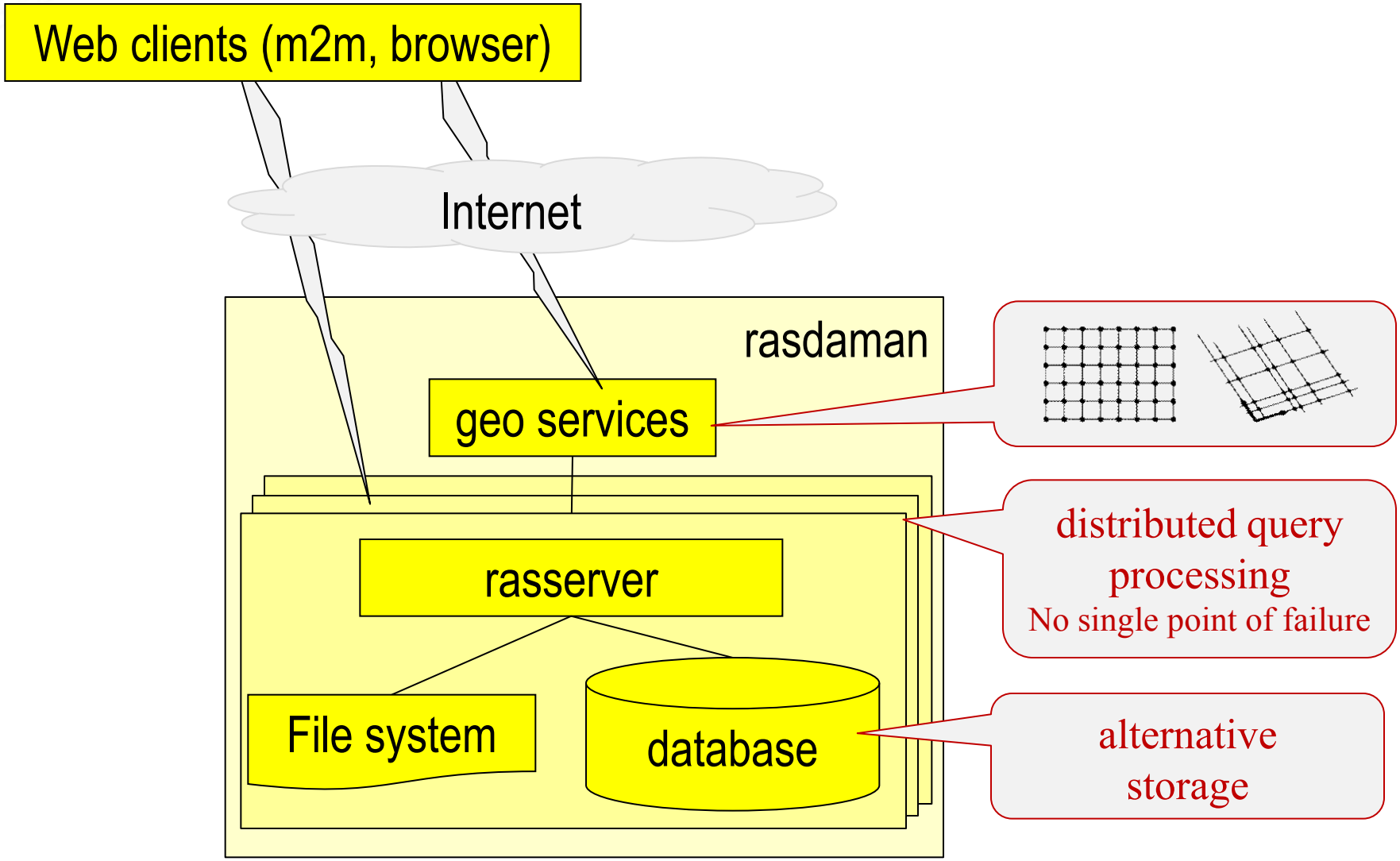
```
select marray i in [0:m], j in [0:p]
       values condense +
              over     k in [0:n]
              using    a [ i, k ] * b [ k, j ]
from   matrix as a, matrix as b
```

- Histogram

```
select marray bucket in [0:255]
       values count_cells( img = bucket )
from   img
```

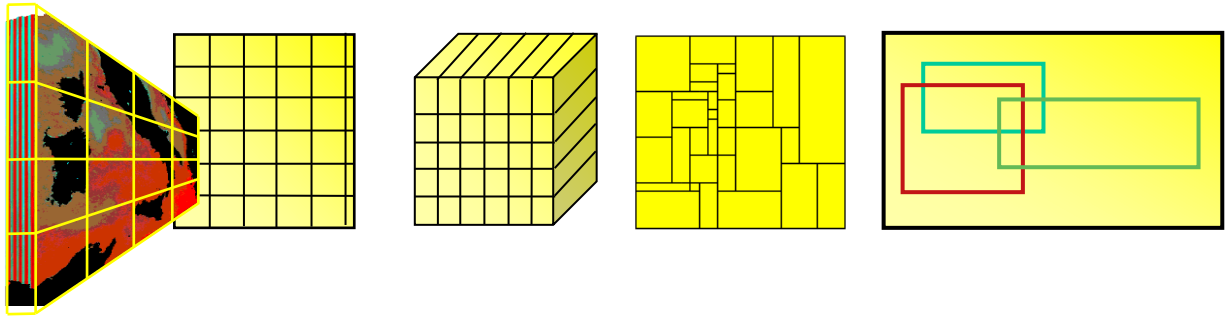


# Architecture



# Adaptive Partitioning („Tiling“)

- Any tiling, canned into strategies [ICDE 1999]
  - *250+ TB datacubes*



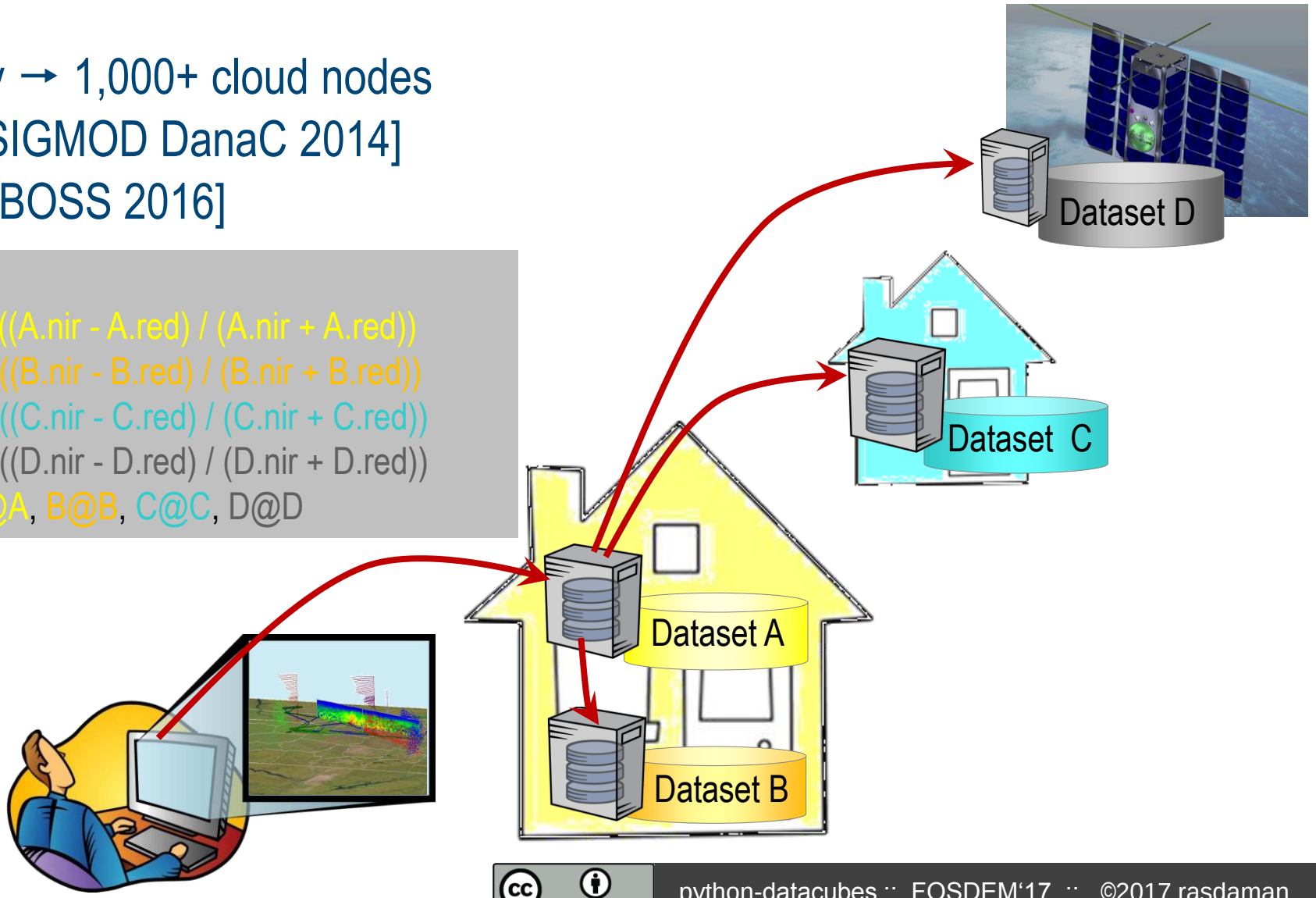
- rasdaman storage layout language [IEEE SSTDM 2010]

```
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
```

# Parallel, Distributed Processing

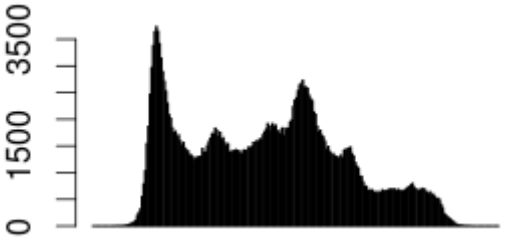
1 query → 1,000+ cloud nodes
[ACM SIGMOD DanaC 2014]
[VLDB BOSS 2016]

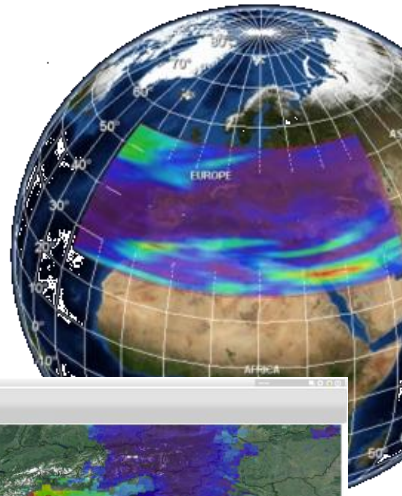
```
select
      max((A.nir - A.red) / (A.nir + A.red))
    - max((B.nir - B.red) / (B.nir + B.red))
    - max((C.nir - C.red) / (C.nir + C.red))
    - max((D.nir - D.red) / (D.nir + D.red))
from A@A, B@B, C@C, D@D
```

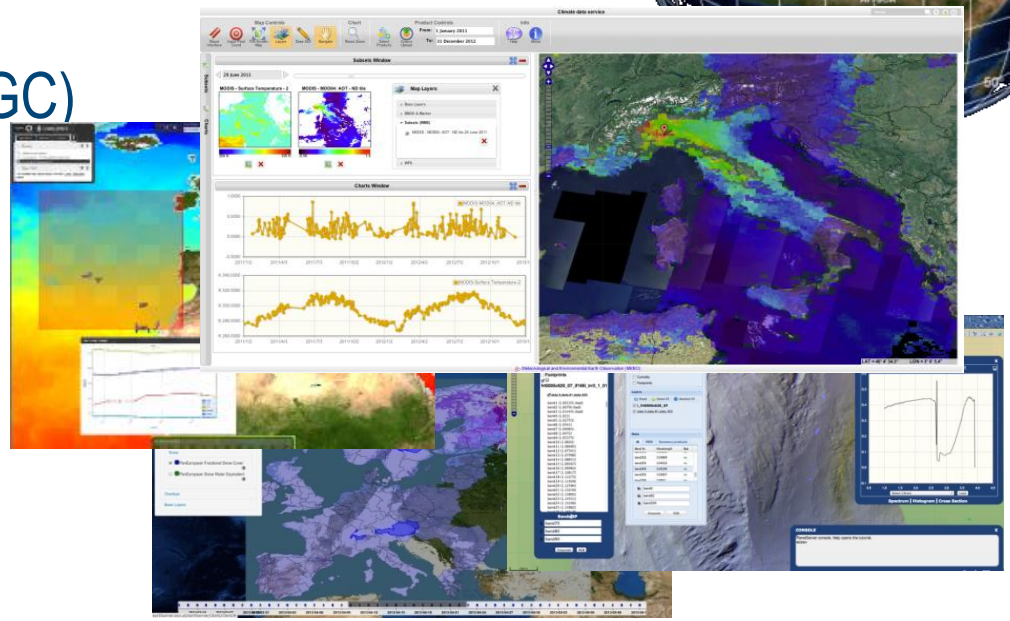


# Science & GIS Tool Interfacing

- General-purpose scientist tools:
  - Java, C++
  - python, R (under work)
- Geo tools:
  - MapServer, GDAL, QGIS, OpenLayers, Leaflet, NASA WorldWind, ...
- Open Geospatial Consortium (OGC) Web Coverage Service (WCS) Core Reference Implementation
  - Can interface to all tools supporting OGC's „Big Geo Data" standards suite



# Python 1: OWSLIB

# OWSLIB

= OGC Web Services Library

- Support for WCS 2 added
  - Olly Clement, PML (Plymouth Marine Laboratory)
  - WCS requests (such as GetCoverage) from python
  - Status: Pull request to be generated soon
- Jupyter notebooks
  - Specifically, OGC WCS from python
  - Julia Wagemann, ECMWF (European Centre for Medium-Range Weather Forecast)
  - Status: Jupyter notebooks online, continuously expanded

# Python 2: general array database backend

MSc thesis by Siddharth Shukla

# Calling rasql from python: Take 1

- Goal: database access
- Basis: numpy, Google GRPC & protobuf
- First approach: call interface

```
from rasdapy.core import Connection
con = Connection()
db = con.database("RASBASE")
txn = db.transaction ()
q = txn.query("SELECT mr from mr")
res = q.eval()

# [ [ . . . ] [ . . . ] [ . . . ] ]
```

# Calling rasql from python: Take 2

- Goal: transparent database access
- Basis: python magic methods
- On top of CLI: python operator overloading
  - Mark object as residing in database
- Lazy evaluation:
  - Collect operations on such object
  - Upon use, generate & execute query
  - Convert result to numpy array
- „Monkey patching“ python operations
  - Ex: „a+b“ in rasql → want database „a+b“ in python
    → overload __add__ magic function, also __radd__, etc.

# Example

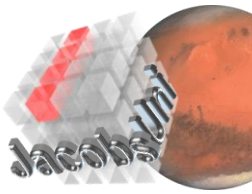
```
>>> from rasdapy.core import Connection
>>> from rasdapy.surface import RasCollection
>>> con = Connection(hostname="127.0.0.1" , port=7001)
>>> mr = RasCollection(con, "mr")
>>> mr = mr[100,150] # Array Subsetting
>>> mr += 1
>>> mr = mr ** 2 # Square of all elements
>>> mr = mr. filter (oid=2)
>>> mr.query
<RasQueryObject>
>>> str (mr.query)
"Select exp(mr[100,150]+1,2) from mr where oid(mr) = 2"
>>> arr = mr.eval ()
<RasArrayObject>
>>> arr.to_array () # Default conversion : Numpy Array
[ [ . . . ] , [ . . . ] , [ . . . ] ]
```

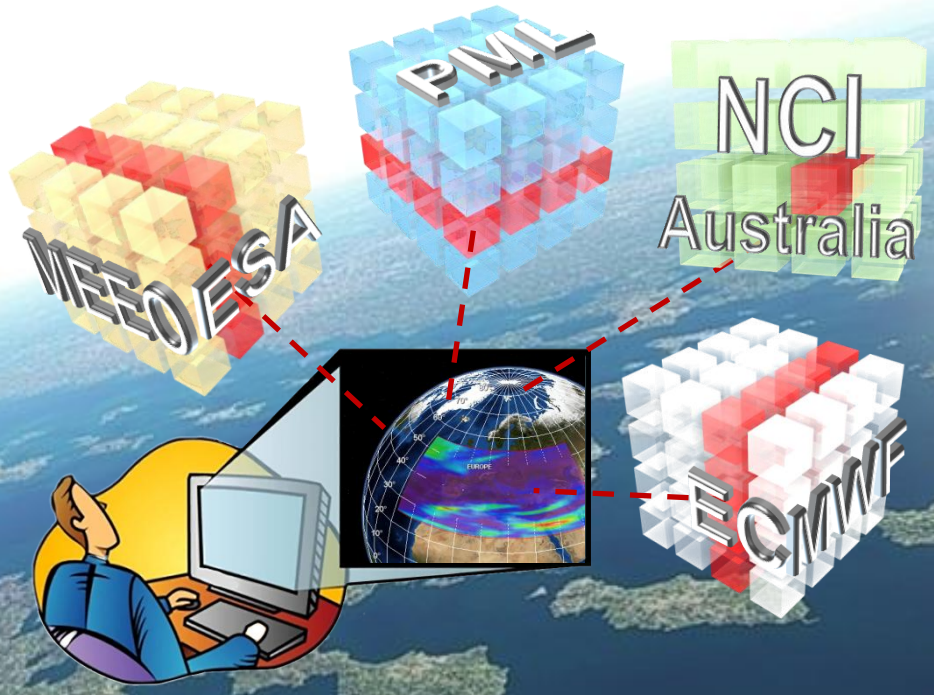
# Summary

rasdaman raster data management | JACOBS UNIVERSITY

# EarthServer: Datacubes At Your Fingertips

- Agile Analytics on x/y/t + x/y/z/t Earth & Planetary datacubes
  - EU rasdaman + NASA WorldWind
  - 100s of TB sites now, next: 1+ Petabyte per cube
- Intercontinental initiative, 3+3 years:
  EU + US + AUS
- Global data federation
  - Access, extract, aggregate, combine any-size datacubes
  - Common basis: OGC WCS





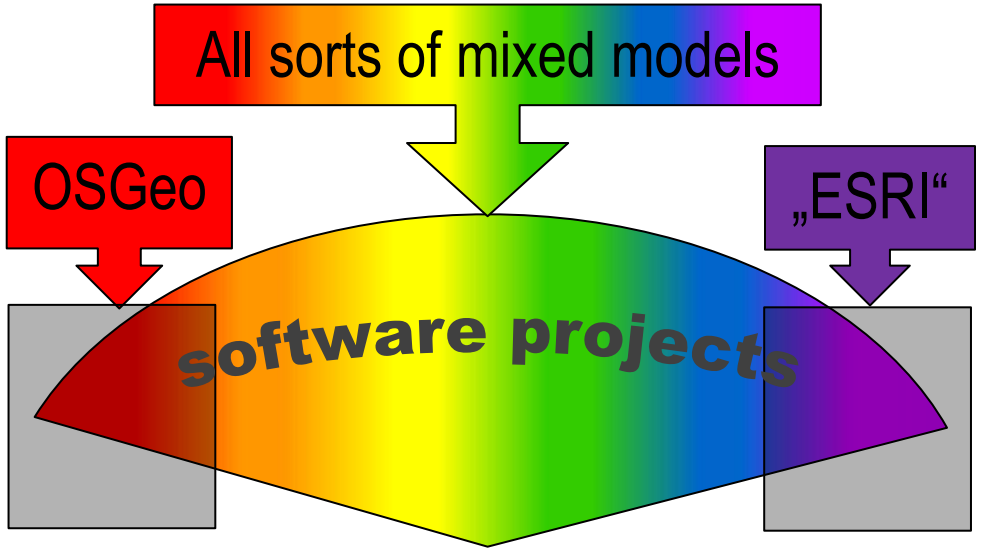

Co-funded by the European Union

www.earthserver.eu

# OSGeo Thoughts: *Representing* Open Source?

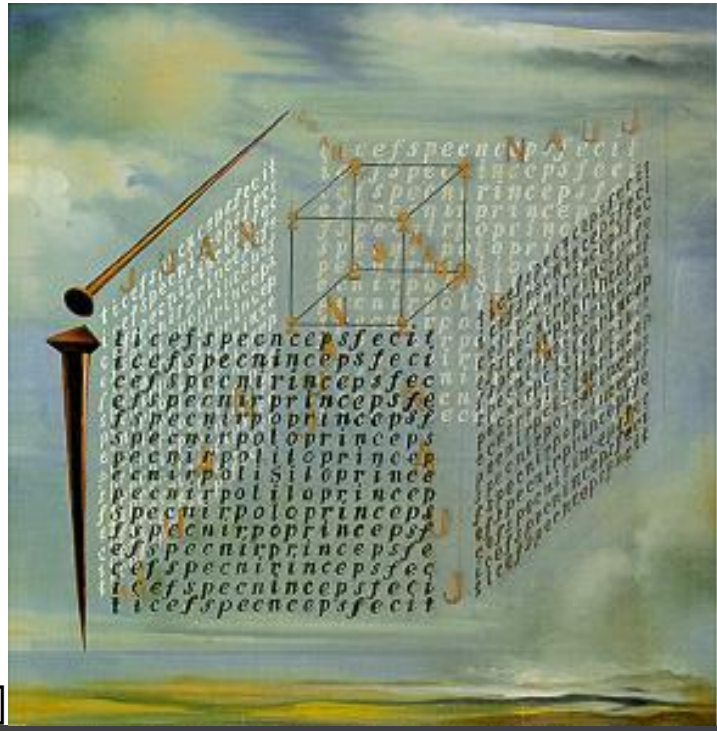
- Organizational Maturity: Process definition & implementation, QM
- Focus on Core Mission
  - Should brand „good software“, not conquer project
  - „design by committee“ over „expert leadership“
- Dogmatic „Software Communism“
  - „all software free“ - why?
  - Large companies don‘t care, small companies vulnerable

  → Dangerous to small enterprises!



# Conclusion

- Array databases for high-level queries on massive n-D arrays
  - QL is good c/s dev interface, but not for (sane) end users
  - rasdaman community: scalable array engine
- Goal: Python, common scientific interface, as gateway to array databases
  - QL transparent
- Status: beta release, to be finished
- See us:
  - www.earthserver.eu
  - www.rasdaman.org
  - www.jacobs-university.de/lsis



[Dali]