Large-Scale Earth Science Services: A Case for Databases

Peter Baumann

International University Bremen
Campus Ring 12, D-28759 Bremen
p.baumann@iu-bremen.de
http://www.faculty.iu-bremen.de/pbaumann

Abstract. Earth sciences are about to follow the mapping domain where raster data increasingly get integrated in online services, contribute by far the largest volume. Interestingly, although more and more raster services are getting online, there is few work on a comprehensive model of raster services, rather architectures are of ad-hoc style and optimized towards very narrow applications, such as fast zoom and pan on seamless maps.

We claim that databases introduce a new quality of service on high-volume multi-dimensional earth science raster data, characterized by clear and understandable concepts, extensibility, and scalability. To support this, we present a comprehensive conceptual model for raster data in earth science and discuss how an efficient architecture can be derived from it, which is implemented in the rasdaman system. Further, we show how such concepts play a role in the development of OGC's geo raster services, WCS and WCPS. Finally, we discuss some research challenges.

Keywords: Raster service, coverage service, rasdaman, OGC.

1 Motivation

Raster data recently increasingly receive attention not only for scientific applications, but even for everyday's convenience such as Internet map services. Advances in storage technology, processing power, and data availability make online navigation on large data volumes feasible. Hence, more and more remote sensing image services are getting online, however usually based on ad hoc implementations. Interestingly there is few work on a comprehensive theory of raster services, rather architectures are optimized towards very narrow applications, such as fast zoom and pan on seamless maps.

In this contribution we claim that database concepts and methods can contribute to an increased quality of service which is characterized by clear (hence, easy to handle) concepts, extensibility, and a potential for high-performance implementations. This opens up new avenues into online data analysis and, generally, advanced services.

In Section 2 we describe the state of the art. Our viewpoints are illustrated through the rasdaman raster middleware for retrieval of n-D raster data stored in relational

J.F. Roddick et al. (Eds.): ER Workshops 2006, LNCS 4231, pp. 75 – 84, 2006.
© Springer-Verlag Berlin Heidelberg 2006

databases [2, 17, 22] in Section 3. In Section 4 we report the state of standardization in the field of geo raster services looking at OGC's Web Coverage Service (WCS) and Web Coverage Processing Service (WCPS). Section 5 concludes the plot.

2 State of the Art

Traditionally, raster data have been stored in sets of files. As the file system has no idea about the semantics (pixel type, number of dimensions, etc.), all selection and processing is burdened to the application developer, leading to tedious, repetitive work and an ill-defined consistency state of the data. Moreover, file-based storage tends to favour particular access patterns (e.g., *x/y* selection on time series) while conveying disastrous performance on all others (e.g., *z/t* selection).

For fast zoom and pan on mosaicked image file sets, many products are available. Access is done through low-level API libraries instead of high-level, model-based query support with internal optimisation and without flexible image extraction functionality, such as hyperspectral channel extraction, overlaying, and *ad hoc* thematic colouring. Most important, file-based solutions *per se* do not scale very well and are inflexible wrt. new requirements. Optimisations done to speed up performance consist in adopting, not to say: re-inventing, one or the other of the set of techniques known in the database community since long – for example, spatial indexing, load balancing, preaggregation, and materialized views.

Relational DBMSs, designed to scale well indeed, traditionally store multi-dimensional arrays as unstructured BLOBs ("binary large objects") introduced by Lorie as "long fields" [12]. This technique cannot give any support for operations beyond line-by-line access, something clearly not feasible for large archives. Tiling, a technique stemming from imaging, has therefore been introduced to databases by rasdaman [1] and recently has been adopted by ESRI's ArcSDE [7].

Object-relational database systems (ORDBMSs) allow to add new data types, including access operations, to the server [20]; examples for such a data type is Oracle GeoRaster [15]. Arrays, however, are not a data *type*, but a data type *constructor* ("template"), parametrized with cell type and dimension – see Section 3.1. Such templates are not supported by ORDBMSs, hence a separate data type has to be defined for 2-D grayscale ortho images, 2-D hyperspectral MODIS images, 4-D climate models, etc. Furthermore, server internal components are not prepared for the kind of operations occurring in MDD applications, therefore important optimization techniques like tile-based pipelining and parallelization of array query trees are difficult to implement.

A literature review specifically on raster databases has been conducted in [21]. Interesting research focuses on specific raster database aspects, such as mass storage support for extreme object sizes [19, Reiner-02] and data models [11, 13]. To the best of our knowledge, rasdaman currently is the only system which combines a formal framework, a declarative, optimizing query language, a system architecture streamlined to large n-D raster objects, and an implementation that is in commercial use.

3 The Rasdaman Raster Server

The rasdaman[1] system has evolved from a series of research projects. Based on an algebraic foundation inspired by the AFATL Image Algebra [18] it provides a raster query language based on SQL92; storage [22, 8] and query optimization likewise are grounded algebraically [2].

Conceptual Model. The conceptual model of rasdaman centers around the notion of an n-D array (in the programming language sense) which can be of any dimension, spatial extent, and array cell type. Following the relational database paradigm, rasdaman also supports sets of arrays. Hence, a rasdaman database can be conceived as a set of tables where each table contains a single array-valued attribute, augmented with an OID system attribute. As rasdaman is domain-neutral, its semantics does not include geo coordinates; this is to be maintained by an additional layer on top of rasdaman.

Arrays can be built upon any valid C/C++ type, be it atomic or composed, based on the type definition language of the ODMG standard [4]. Arrays are defined through a template `marray<b,d>` which is instantiated with the array base type `b` and the array extent (*spatial domain*) `d`, specified by the lower and upper bound for each dimension (which can be left open for variable arrays). Thus, an unbounded colour ortho image can be defined by

```
typedef marray
<   struct{ char red, green, blue; }, [ *:*, *:* ]
> RGBOrthoImg;
```

Type definitions serve for both semantic checks and optimization during query evaluation [17].

Array Retrieval. The rasdaman query language, rasql, adds n-D raster expressions to ISO SQL92. Like SQL, a rasql query returns a set of items (in this case, raster objects or metadata information). *Trimming* produces rectangular cut-outs, specified through the corner coordinates; a *section* produces a slice with reduced dimensionality.

Example 1: "A slice at time *t* through *x/y/t* cube SatTimeSeries, cutting out the area between (*x0*,*y0*) and (*x1*,*y1*)":

```
select SatTimeSeries[x0:x1,y0:y1,t]
from   SatTimeSeries
```

For each operation available on the cell (i.e., pixel) type, a corresponding *induce operation* is provided which simultaneously applies the base operation to all raster cells. Both unary (e.g., record access) and binary operations (e.g., masking and overlaying) can be induced.

Example 2: "Color ortho image Ortho, overlaid with bit layer 3 of thematic map TMap coloured in red":

```
select Ortho overlay bit( TMap, 3 ) * {255c,0c,0c}
from   TMap, Ortho
```

[1] "raster data manager", www.rasdaman.com

In general, raster expressions can be used in the `select` part of a query and, if the outermost expression is of type Boolean, also in the `where` part [2].

Condense operations derive summary data. The general condenser is of the form

```
condense op
over     x in dom
using    expr
```

This expression will iterate over domain `dom`, binding variable `x` to each location in turn and evaluating `expr`, which may contain occurrences of `x`; all evaluation results are combined through binary operation `op` which must be commutative and associative so that evaluation sequence can be chosen by the server. Shorthands are available for the frequently used special cases sum, max, min, average, and for boolean quantifiers, similar to the SQL aggregate functions.

Example 3: "For all ortho images Ortho, the ratio between min and max in the near infrared component":

```
select min_cells(Ortho.nir) / max_cells(Ortho)
from   Ortho
```

Finally, the *marray* constructor allows to establish a new raster structure, possibly derived from an existing one. Following the syntax

```
marray x in dom
values expr
```

the query engine creates a new array of domain dom and cell type identical to the cell type of expression expr; then, it iterates over domain binding variable x to each location in turn, evaluates expression expr (which may contain occurrences of x), and finally assigns the result to the resp. cell in the result.

Example 4: "Histogram over the red channel of Ortho ":

```
select marray bucket in [0..255]
       values add_cells( Ortho.red = bucket )
from   Ortho
```

The result raster object is a 1-D array containing 256 entries for all possible 8-bit intensity values. The induced comparison of the red channel with scalar value `bucket` yields a boolean result which is interpreted as 0 or 1, resp., and then summed up for each bucket.

Actually, general condenser plus operator are sufficient to describe all rasql raster operations; induce, trim, section operations etc. are just shorthands for convenience. This narrow basis significantly eases both conceptual treatment and implementation.

The expressiveness of rasql enables a wide range of signal processing, imaging, and statistical operations up to, e.g., filter kernels. The expressive power tentatively has been limited to non-recursive operations, thereby guaranteeing termination of any well-formed query; this principle is known as “safe evaluation” in query languages.

Array Storage. Raster objects are maintained in a standard relational database, based on the partitioning of an raster object into *tiles* [8]. Aside from a regular subdivision, any user or system generated partitioning is possible (Fig. 1); several different

strategies are available. A geo index is employed to quickly determine the tiles affected by a query. Optionally tiles are compressed using one of various choices, using lossless or lossy (wavelet) algorithms; independently, query results can be compressed for transfer to the client. Both tiling strategy and compression comprise database tuning parameters.

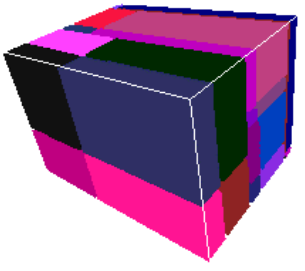

Fig. 1. Tiled 3-D raster object

Tiles and index are stored as BLOBs in a relational database which also hold the data dictionary needed by rasdaman's dynamic type system. Adaptors are available for several relational systems, among them open-source PostgreSQL. Interestingly, PostgreSQL has shown adequate to serve multi-Terabyte ortho imagery under industrial conditions.

Query Evaluation. Queries are parsed, optimised, and executed in the rasdaman server. The parser receives the query string and generates the operation tree. Further, it applies algebraic optimisation rules to the query tree where applicable [17]; of the 150 algebraic rewriting rules, 110 are actually optimising while the other 40 serve to transform the query into canonical form. Parsing and optimization together take less than a millisecond on a PC.

Query execution is parallelised. First, rasdaman offers inter-query parallelism: A dispatcher schedules requests into a pool of server processes on a per-transaction basis. Ongoing research work involves intra-query parallelism where the query tree transparently is distributed across available CPUs or computers in the network so that each CPU evaluates a subset of the tiles [9]. Preliminary performance results are promising, showing speed-up / #CPU ratios of 95.5%.

For arrays larger than disk space, hierarchical storage management (HSM) support has been developed [16].

Performance. As benchmarks show [3], databases can not only compete with, but outperform file-based raster services. Several factors contribute to this:

- Optimized tiling makes the server fetch much less data from disk. E.g., 4-D climate models are generated and stored in *x/y* slices, while evaluation often is along the time axis. Similar effects occur with remote sensing time series sliced along *t*.
- Enabling the client to send one complex request instead of a long sequence of atomic operations opens up space for the optimizer. Query optimization particularly pays off then. Even in a "simple" WMS request containing overlays and layer coloring the rasdaman optimizer can achieve high gain.
- Further, by offering complex operations to the client there is much less communication overhead: no intermediate results have to be transferred back and forth, and the final result is exactly what the client needs, hence the minimum amount of data that is required to answer the client's needs (cf. Fig. 2 right).

In summary, our observation is that databases can outperform file-based approaches whenever flexibility for complex ad-hoc requests and high scalability is required.

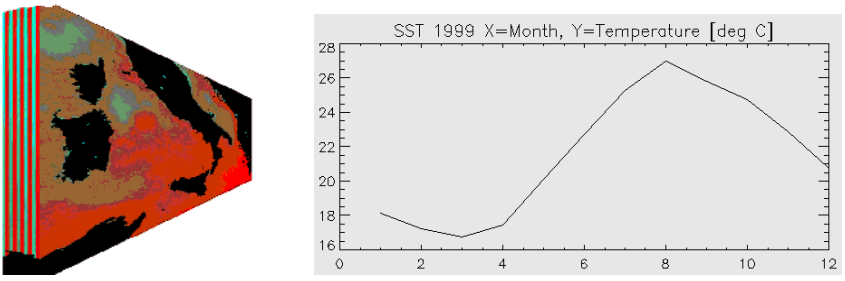

Fig. 2. DLR AVHRR SST time series service: 3-D trim around Italy, Sicily, and Corsica (left) and 1-D section obtaining the temperature curve for a chosen location. Data obtained from the rasdaman/Oracle server through WCS.

Status. The rasdaman system is in operational use since many years both in mapping agencies, in (mining) industry, and in research. Fig. 2 shows a cutout from a rasdaman/Oracle database where about 10,000 images have been merged into a seamless 3-D *x/y/t* cube of AVHRR sea-surface temperature imagery. Implemented in 2002, it uses an early-version WCS interface. The experience gained there now is being exploited in OGC for raster geo service standardization.

4 Geo Raster Service Standardization

The main driver for interoperable geo service standards is the Open GeoSpatial Consortium (OGC). The Web Coverage Service (WCS) which specifies retrieval of all or part of a coverage object; although coverages have a more general definition, WCS currently supports only regularly gridded rasters. Based on the WCS model, the Web Coverage Processing Service (WCPS) extends WCS with a coverage expression language allowing requests of unlimited nesting complexity. Having two different standards, WCS and WCPS, increases modularity: implementers can choose to implement only the – relatively simple – WCS, or to undertake a WCPS implementation, which is more challenging.

4.1 WCS

Based on the coverage definitions of ISO 19123 [10], WCS specializes on retrieval from coverages, with emphasis on providing available data together with their detailed (metadata) descriptions and returning data with its original semantics (instead of pictures, like, e.g., WMS).

In WCS 1.1, a coverage is seen as a 2-D, 3-D, or 4-D spatio-temporal matrix (aka tensor). The coverage extent is called its *domain*, the coverage value data type comprises the coverage *range*. A range definition consists of a list of fields; a field is either atomic (such as temperature), or compound, in which case it consists of an n-D tensor which can be addressed along the axes (in WCS speak: keys chosen from keylists), very much like array indexing.

To accommodate orthorectified/georeferenced, georeferenced, and non-georeferenced imagery, a coverage can bear either a ground Coordinate Reference System (CRS), or an image CRS, or both. Addressing is possible via either CRS. A bounding box is associated with each coverage, expressed in the resp. CRS(s).

Operationally, a WCS client first obtains information about the service and coverage offerings; Subsequently, coverages or subsets thereof can be retrieved using a `GetCoverage` request. Requests and responses are exchanged via http, using either key/value pair (KVP) or XML encoding; the XML structures are laid down using XML Schema. `GetCoverage` offers a set of six operations to be executed on a coverage, controlled by the request parameters: spatio-temporal domain subsetting; range subsetting (aka "band" selection, addressing via fields and their keys, if defined); resampling (e.g., scaling); reprojection; data format encoding; and result file(s) delivery; results can optionally be delivered directly in the GetCoverage response, or stored by the server for later download by the client.

4.2 Web Coverage Processing Service (WCPS)

The Web Coverage Processing Service (WCPS) Implementation Specification has been designed to support non-trivial navigation and analysis on large-scale, multi-dimensional sensor and image data. In the sequel we outline the WCPS concepts, see [14] for details.

Coverage Model. WCPS grounds in its coverage model on WCS, with one extension: coverages can have not only spatio-temporal, but also abstract axes[2]. Examples of such abstract axes are simulation time for climate model computations, input parameter spaces for climate models, or statistical feature spaces. This semantics is known to the server; for example, operations that refer to geo coordinates – such as reprojection – can only be applied to *x* and *y* axes. All axes are treated equally in the operations; for example, subsetting and slicing operations can be performed on every axis.

The coverage locations containing values are referred to as *cells*; the (atomic or composite) values associated with a particular cell are called its *cell values*.

Operational Model. The WCPS request structure corresponds to WCS, with only some technically motivated extensions. In place of the WCS `GetCoverage` request, WCPS offers `ProcessCoverage` where the retrieval expression is passed to the server. The WCPS language is defined as an abstract language, with mappings to both key-value pair (KVP) and XML encodings.

A WCPS `ProcessCoverage` request consists of a central request loop where each of a list of coverages in turn is visited to instantiate the associated processing expression if an optionally provided predicate is fulfilled. Coverage names refer to the the list advertised by the service. WCPS Abstract Syntax for the request loop is

[2] This feature is being discussed in the WCS group for possible inclusion in WCS 1.2.

```
for c in ( coverageList )
[ where cond(c) ]
return pExpr(c)
```

Variable *c* iterates over the coverages enumerated in *coverageList*, considering only those where the predicate `cond(c)` is fulfilled. The processing expression `pExpr(c)` consists of either a metadata accessor operation (such as `tdom(c)` returning `c`'s temporal domain extent), or of an encoding expression

```
encode(e,f)
```

where `e` is a coverage-valued expression and `f` is the name of a supported format.

Example: "*Coverages A, B, and C, TIFF-encoded*":

```
for c in ( A, B, C )
return encode( c, "tiff" )
```

WCPS operations are very similar in nature to rasql, extended with geo-specific functionality like mapping of spatial and temporal coordinates into cell coordinates and reprojection. For example, if a client wishes to express slicing in spatio-temporal coordinates, it needs to explicitly apply the coordinate mapping function `ttransform()`.

Example: "*Slice through A at time 'Thu Nov 24 01:33:27 CET 2005'* ":

```
for c in ( A )
return encode( sect(c,3, ttransform(c,
            "Thu Nov 24 01:33:27 CET 2005")), "tiff" )
```

This may seem complex and unwieldy. However, this allows addressing on both geo and pixel level; further, this language is not intended for humans – rather, some GUI client will offer click-and-point request composition and then internally generate and ship the corresponding XML request.

Like in rasql, *induce expressions* allows to simultaneously apply a cell operation to a coverage as a whole.

Example: "*Sum of red and near infrared band from coverage A, as 8-bit integer*":

```
for c in ( A )
return encode( (char) c.red+c.nir, "tiff" )
```

Operations involving interpolation allow to optionally indicate an interpolation method.

Example: "*Coverage A, scaled to (a,b) along the time axis*"; this assumes that the server offers cubic time interpolation on this coverage:

```
for c in ( A )
return encode( scale(c,t,a,b,"cubic"), "tiff" )
```

The encoded result coverage(s) are, at the client's discretion, either shipped back immediately as XML response or stored at the server for some time and only the URL is returned. Scalar results (like condensers and metadata) are returned immediately.

The reference implementation currently under [5, 6] way maps WCPS requests received to rasdaman queries, lets rasdaman compose the result coverage(s), and finally adds the XML response metadata. Implementation language is Java.

5 Conclusion

All across the earth sciences the vaults of huge raster data archives need to be opened for scientific exploitation. Retrieval technology for large raster repositories, however, is lagging behind. Raster archives today commonly are implemented in a file-based manner; databases serve only for meta data search, but not for image retrieval itself. Hence, storage usually is driven by the data acquisition process rather than by user access patterns, resulting in inefficient access. Scalability often is achieved only through massive hardware support. Moreover, versatile retrieval as known from database query languages for alphanumeric data is impossible, not to speak of other services like transactions.

Aside from flexibility in task definition, there are several more arguments which advocate the use of database systems. Query languages allow to define complex tasks to the server rather than a small set of atomic steps as in procedural APIs; the consequence is that the query optimiser gains a lot of freedom to rephrase the query optimally for the particular situation. Further, application integration is much higher because there is one central instance in charge of data integration and consistency. All in all, file-based solutions frequently re-invent all the features which have been developed by database technology over decades; using existing mature technology obviously is preferable.

The new generation of geo service standards taps into experience from different domains, among them databases. WCPS is an example where a coverage expression language is defined in a way such as to allow open-ended functionality through few, well-defined concepts which are abstract enough to allow for many transparent server-side optimizations.

WCS 1.1 is expected to be released in Fall 2006. WCPS currently is OGC approved Best Practices Paper and is expected to become Draft Standard in Fall/Winter 2006.

Current work encompasses upgrading of the rasdaman-based WCS 1.0 implementation to the forthcoming version 1.1, and finalizing the WCPS reference implementation, including the server [5], a visual programming client [6] and a compliance test suite.

Acknowledgements

The author is indebted to John Evans (NASA; WCS co-editor) and Arliss Whiteside (BEA Systems; WCS co-editor); Steven Keens (PCI Geomatics); Luc Donea (Ionic); Peter Vretanos (Cubewerx); Wen-Li Yang (NASA); and all the other WCS Revision Working Group members; particular thanks goes to Sean Forde (Lizardtech; Coverages WG speaker).

Further, many thanks to Georgi Chulkov and Ivan Delchev for their great work implementing WCS/WCPS server and clients, and for patiently exploring all the conceptual pitfalls.

References

1. Baumann, P.: Database Support for Multidimensional Discrete Data. Proc. 3rd Intl Symp. on Large Spatial Databases, LNCS 692, Springer 1993, pp. 191-206
2. Baumann, P.: A Database Array Algebra for Spatio-Temporal Data and Beyond. Proc. Next Generation IT and Systems (NGITS), Zikhron Yaakov, Israel, 1999, pp. 76 – 93
3. Baumann, P.: Web-enabled Raster GIS Services for Large Image and Map Databases. Special Track on Image-Based Geospatial Databases, 5th Int'l Workshop on Query Processing and Multimedia Issues in Distributed Systems (QPMIDS'2001), September 3-4, 2001, Munich, Germany
4. Cattell, R.G.G.: The Object Database Standard: ODMG 2.0. Morgan Kaufmann Publishers, California, 1997
5. Chulkov, G.: Architecture and Implementation of a Web Coverage Processing Service Using a Database Back-End. IUB Bachelor's Thesis, May 2006
6. Delchev, I.: Graphical Client for a Multidimensional Geo Raster Service. IUB seminar report, May 2006
7. n.n.: Raster Data in ArcSDE 9.1. ESRI White Paper, September 2005
8. Furtado, P. , Baumann, P.: Storage of Multidimensional Arrays Based on Arbitrary Tiling. Proc. ICDE, Sydney, Australia, 1999, pp. 480-489
9. Hahn, K., Reiner, B., Höfling, G., Baumann, P.: Parallel Query Support for Multidimensional Data: Inter-object Parallelism. Proc. DEXA, Aix-en-Provence, France, 2002.
10. n.n.: FDIS 19123 Geographic information - Schema for coverage geometry and functions. ISO/TC 211, document 1740, October 2004
11. Libkin, L., Machlin, R., Wong, L.: A Query Language for Multidimensional Arrays: Design, Implementation, and Optimization Techniques, Proc. ACM SIGMOD, Montreal, Canada, 1996, pp. 228 – 239
12. Lorie, R.A.: Issues in Databases for Design Transactions. in: Encarnaçao, J., Krause, F.L. (eds.): File Structures and Databases for CAD, North-Holland Publishing, 1982
13. Marathe, A.P., Salem, K., Query Processing Techniques for Arrays. Proc. ACM SIGMOD '99, Philadelphia, USA, 1999, pp. 323-334
14. OGC: Web Coverage Processing Service. OGC Best Practices Paper, artifact 06-102, August 2006, available from OGC Portal (www.opengis.org)
15. n.n.: Oracle Database 10g: Managing Spatial Raster Data using GeoRaster. Oracle Technical Whitepaper, May 2005
16. Reiner, B., Hahn, K., Höfling, G., Baumann, P.: Hierarchical Storage Support and Management for Large-Scale Multidimensional Array Database Management Systems. Proc. DEXA, Aix en Provence, France, 2002
17. Ritsch, R.: Optimization and Evaluation of Array Queries in Database Management Systems. PhD Thesis, Technische Universität München, 1999.
18. Ritter, G., Wilson, J., Davidson, J.: Image algebra: An Overview. Computer Vision, Graphics, and Image Processing, 49(1):297-331; 1990
19. Sarawagi, S., Stonebraker, M.: Efficient Organization of Large Multidimensional Arrays.. Proc. ICDE'94, Houston, USA, 1994, pp. 328-336
20. Stonebraker, M., Moore, D., Brown, P.: Object-Relational DBMSs: Tracking the Next Great Wave (2nd edition). Morgan Kaufmann, September 1998
21. Varsandan, I.: State of the Art: Arrays in Databases. IUB Bachelor's Thesis, May 2006
22. Widmann, N.: Efficient Operation Execution on Multidimensional Array Data. PhD Thesis, Technische Universität München, 2000