# Spatiotemporal Data-Cube Retrieval and Processing with xWCPS

George Kakaletris[1], Panagiota Koltsida[2], Manos Kouvarakis[1] and Konstantinos Apostolopoulos[1]
[1]*Communications & Information Technologies Experts S. A. Athens, Greece*
[2]*Department of Informatics & Telecommunications, University of Athens, Athens, Greece*

Keywords: Query Language, Array Databases, Coverages, Metadata.

Abstract: Management and processing of big data is inherently interweaved with the exploitation of their metadata, also "big" on their own, not only due to the increased number of datasets that get generated with continuously increased rates, but also due to the need for deeper and wider description of those data, which yields metadata of higher complexity and volume. Taking into account that generally data cannot be processed unless enough description is provided on their structure, origin, etc, accessing those metadata becomes crucial not only for locating the appropriate data but also for consuming them. The instruments to access those metadata shall be tolerant to their heterogeneity and loose structure. In this direction, xWCPS (XPath-enabled WCPS) is a novel query language that targets the spatiotemporal data cubes domain and tries to bring together metadata and multidimensional data processing under a single syntax paradigm limiting the need of using different tools to achieve this. It builds on the domain-established WCPS protocol and the widely adopted XPath language and yields new facilities to spatiotemporal datacubes analytics. Currently in its 2nd release, xWCPS, represents a major revision over its predecessor aiming to deliver improved, clearer, syntax and to ease implementation by its adopters.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

## 1 INTRODUCTION

Petabytes of data of scientific interest are becoming available as a result of humanity's increased interest and capability to monitor natural processes but also to model them and explore the results of those theoretical models under different conditions. This trend on its own puts data infrastructures storage and transfer mechanisms under severe pressure, not to mention the processing ones. Duplicating or moving those data at their final consumption point is usually beyond its capabilities, as network and local storage capabilities cannot catch up this trend. It is a direct consequence of this observation that it is important to develop mechanisms to support efficient data identification, filtering and in-situ processing that will reduce the need for unnecessary data move and duplication.

On the other hand, it is also evident that in order to pick the appropriate data and subsequent to consume them, one needs to be able to identify those data among a huge number of data sets. This sums to the point that more complex and more detailed metadata that cover an increasing number of aspects of the data they describe are required and produced.

As the volume of produced data grows larger, so does the volume of metadata that offer information about them, and it is evident that their handling need efficient retrieval mechanisms too, however those can no longer be considered independently of the mechanisms that handle the data, as both are needed together. A data science field where those observations apply to their full extent, and the area where the work presented herein focuses on, is the geospatial one. Data from the domain are multidimensional, diverse in terms of content and size and are accompanied with metadata that are essential for their retrieval and processing.

Regarding data management, traditional database management systems (DBMSs) do not efficiently support array data, which is the most common form of data met here. This led to the development of dedicated array DBMSs like SciDB (Brown, 2010) and Rasdaman (Baumann, 1998), which close this gap by extending the supported data structures with multidimensional arrays of unlimited size, thus enabling the efficient storage of spatiotemporal data. Although array databases manage to handle these types of data they lack dealing with metadata filtering and processing in a unified way.

Kakaletris, G., Koltsida, P., Kouvarakis, M. and Apostolopoulos, K.
Spatiotemporal Data-Cube Retrieval and Processing with xWCPS.
DOI: 10.5220/0006814601480156
In *Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2018)*, pages 148-156
ISBN: 978-989-758-294-3
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

Our approach, manages to deliver efficient cross disciplinary querying and processing of array data and metadata, by offering a unified and friendly way through the xWCPS 2.0. xWCPS 2.0 is built on the first specification of the language xWCPS 1.0 (Liakos, 2014), as defined in EarthServer Project (EarthServer.eu, 2018) and refines its characteristics, so it facilitates implementation, improves expected query performance and eases user adoption and usage. In contrast to traditional approaches, where two different queries are required so as to first filter the semi structured metadata, retrieve and process the results, in our approach the same functionality can be achieved by executing just one unified query, resulting the least number of data transferred to their final consumption point. To overcome those limitations, we propose a query language with clear and user friendly syntax and we offer a working engine for it, in which its core components are a new metadata management engine, called FeMME, that follows a scalable no-SQL approach fitting the needs of the endeavour, and a proven array database system, Rasdaman and we efficiently combine them to support unified processing and retrieval of array data and metadata.

## 2 CONCEPTS AND MOTIVATION

The fundamental ideas behind this work have emerged from the EarthServer project series, which set as an objective to establish Agile Analytics on Petabyte Data Cubes as a simple, user-friendly and scalable paradigm. The mandate of the project includes the delivery of a standards' based, declarative query language that enhances geospatial data infrastructures by allowing combined multidimensional data and metadata filtering and processing. xWCPS, in its current 2$^{nd}$ version, is built on top of xWCPS 1.0 and combines two widely known specifications, XPath (W3c.org, 2018) and the Web Coverage Processing Service (WCPS) standard (Baumann, 2010) into a single FLWOR (acronym For-Let-Order By-Where-Return, which stands for For-Let-Where-Order-Return) syntax to achieve the aforementioned result.

In the root of the overall approach lies the concept of "coverage" (OGC, 2017), a fundamental element in Open Geospatial Consortium (http://www.open geospatial.org/) ecosystem. The coverage refers to data and metadata representing multidimensional space/time-varying phenomena. The OGC has introduced a number of standards and specifications for accessing, retrieving and processing coverages, the Web Coverage Service (WCS) (Baumann, 2012) being one standard to support the access of raster data that are handled as coverages. WCS defines requests against these data and returns data with original semantics (instead of raster images). WCS supports the delivery of rich metadata about coverages, however it yields huge flexibility on those metadata to their provider making assumptions on the nature and form of those metadata, irrelevant. Complementing WCS, the Web Coverage Processing Service offers processing capabilities on top of array data using its defined language, allowing ad-hoc processing of coverage data. Examples include deriving composite indices (e.g. vegetation index), determining statistical evaluations, and generating different kinds of plots like data classifications, histograms, etc.

The primary structure of the WCPS language comprises the for-where-return clauses. The "for" clause specifies the set of coverages that will be examined by a query. The "return" clause specifies the potential output that may be appended to the list of results, in each iteration defined by the "for" clause. Criteria used for determining if the output of "return" is actually appended are specified by the "where" clause.

It has to be noted that WCS and WCPS are implemented by Rasdaman array database, which is OGC's official Reference Implementation for WCS Core. Rasdaman (Baumann, 1998), raster data manager, is a fully parallel array engine that allows storing and querying massive multi-dimensional arrays, such as sensor data, satellite imagery, and simulation data appearing in domains like earth, space, and life sciences. This array analytics engine distinguishes itself by its flexibility, performance, and scalability. From simple geo imagery services up to complex analytics, Rasdaman provides the whole spectrum of functionality on spatio-temporal raster data - both regular and irregular grids.

Although in EarthServer project they are faced from the geospatial domain standpoint and expressed as coverages, multi-dimensional arrays are far from domain specific data form, and play a central role in all science, engineering, and beyond. Consequently, a significant number of approaches for retrieval from arrays have been proposed for different purpose applications. In practice, though, arrays typically are forming part of some larger data structure. Array SQL is a horizontal technology that – by its key enabling features *flexibility*, *scalability*, and *information integration* – enhances all fields of data management.

In this domain it is evident that data cannot be consumed without metadata describing their essence.

Various and important characteristics reside into their metadata, thus making the consideration of joint filtering and processing of data and metadata a fundamental requirement. However, metadata engagement in this context has been largely ignored until recently, and this is the gap that our approach comes to fill in.

In prior approach, in order to accommodate this requirement, xWCPS 1.0 (Liakos et Al, 2014) has been specified and implemented by merging the WCPS standard with the XQuery language, thus eliminating the limitations of the WCPS and WCS queries and allowing the parallel and combined query and processing for both data and metadata. Although xWCPS 1.0 and its initial implementation met the requirements mentioned above and managed to fill the gap of jointly accessing and processing data and metadata, it was evident that would not easily cope with challenges of the near future, where billions of datasets may be present in a federated infrastructure of even in single data server. The main problems of this approach can be summarized below: a) its syntax proved to be cumbersome for the users, especially dealing with the XQuery syntax and b) its engine implementation relied on XML management systems, with full XQuery support, that could not perform as required for extremely large metadata volumes and complexity. These limitations became evident during the adoption tests of xWCPS 1.0 that took place in the EarthServer-1 project.

To overcome all these issues xWCPS 2.0 has been designed and implemented in EarthServer-2 project and is presented in detail in the following sections.

## 3 xWCPS 2.0

The management of big multidimensional datasets, e.g. coverages, poses a number of issues and challenges due to their size, nature and the diversity in phenomena and processes they might represent. Combining this with the velocity those are generated, be it generation of data from sensing, simulations and transformations, the demand for efficiently identifying, filtering and processing them (and if possible in a distributed manner) has emerged. At a certain point where users need to refer to large data stores, it becomes clear that the simultaneous utilization of metadata and array data is required so that the precise piece of data needed is located and processed according to its form and characteristics and the requirements of its consumer/client.

To accommodate this in our approach, two well-known standards, XPath for metadata filtering/extracting and the Web Coverage Processing Service (WCPS) for array data processing, are combined, allowing an operation to be executed utilizing both of them without roundtrips or explicit knowledge of the characteristics of the data and the system they reside on, that is the case until now at least in the geospatial domain. The result is a declarative query language that follows the For-Let-Where-Order-Return paradigm (expressed as FLWOR) that offers a clear, well defined syntax, improving the way scientific data can be accessed and eliminating the need of prior knowledge of the data identifiers and characteristics. A similar approach was followed in the xWCPS 1.0, however use of full XQuery was assumed leading to the no user acceptance due to its bewildering syntax. The 2nd release drops XQuery in favour of XPath, with some additional elements yielding several positive results both from implementation and utilization standpoints. In the rest of this section we provide a brief introduction to the fundamentals of xWCPS 2.0 describing the core idea, its syntax, a number of use cases, and summarizing important notions for this paper. In the rest of the paper, for simplicity, we refer to xWCPS 2.0 with the term xWCPS.

### 3.1 Approach

One of the fundamental operations a query language must offer is that of querying for all data residing in the database without prior knowledge of their internal representation. WCPS requires the specification of coverage identifiers in selection queries. These identifiers are part of the database's resource description and can be retrieved by issuing a WCS operation. This step introduces overhead in the querying process, which significantly constrains the user-friendliness of the query language and undermines the overall user experience. Another very common feature a query language must offer is that of filtering results according to some specified criteria. However, when a user asks for array data with WCPS in order to select coverages, it is not possible to define conditions regarding the accompanying metadata. Finally, yet importantly, it is fundamental for a language to return all the available information, containing both data and metadata.

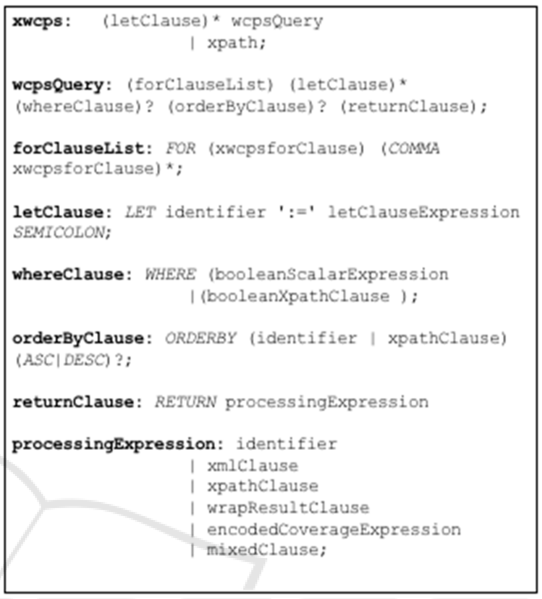
xWCPS (XPath Enabled WCPS) is a Query Language (QL) introduced to fill these gaps, merging two widely adopted standards, namely XPath 2.0 because of its capabilities on XML handling and WCPS's raster data processing abilities, into a new construct, which enables simultaneous exploitation of both coverage metadata and payload in data processing queries. By combining those two, it is delivering a rich set of features that revolutionizes the way scientific data can be located and processed, by enabling combined search, filtering and processing on both metadata and OGC coverages' payload. In brief, queries expressed in xWCPS are able to utilize coverage metadata - commonly expressed in XML - by incorporating support for FLWOR expression paradigm and providing the appropriate placeholders that enable any XPath or WCPS or combined query to be expressed in its syntax.

Expressiveness and coherence are key features of the language, now in its 2nd revision, allowing experts dealing with multidimensional array data to easily adopt and take advantage of its offerings. In general, xWCPS is designed to consist the following features:

- **Coverage Identification based on Metadata:** WCPS requires the specification of coverage identifiers in selection queries. xWCPS is introduced to fill this gap and eliminate the need of prior knowledge of the data by offering a unified interface aiming at being rich, expressive and user friendly and allowing coverage selection based on an XPath expression.
- **Exploitation of Descriptive Metadata:** Coverage filtering based on the available metadata using XPath 2.0. For and where clauses can contain XPath 2.0 expressions in order to restrict results to specific metadata.
- **Repetitiveness Reduction:** xWCPS supports variable manipulation, which allows assigning complex expressions to variables and re-using them for subsequent references, avoiding repetitiveness.
- **Extended Set of Results Support:** An important feature of xWCPS is the ability to return the data accompanied with their metadata.

## 3.2 Syntax

Queries are the most fundamental part of the language. A simple WCPS query is based on a "for-where-return" structure. An xWCPS query is composed from several expressions, including the basic three clauses "for-where-return" of WCPS, while introducing the "let-order by" structure and XPath 2.0. Additionally, xWCPS includes special operators to provide easier search abilities to filter specific metadata. The top-level grammar of xWCPS is presented on Figure 1. xWCPS acts as a wrapper construct on top of XPath 2.0 and WCPS, thus it doesn't offer any language specific operations. Every valid WCPS or XPath 2.0 operation is a valid xWCPS operation; xWCPS combines WCPS with XPath 2.0 operations using a rather simple syntactic formalism.



```
xwcps:   (letClause)* wcpsQuery
                | xpath;

wcpsQuery: (forClauseList) (letClause)*
(whereClause)? (orderByClause)? (returnClause);

forClauseList: FOR (xwcpsforClause) (COMMA
xwcpsforClause)*;

letClause: LET identifier ':=' letClauseExpression
SEMICOLON;

whereClause: WHERE (booleanScalarExpression
                |(booleanXpathClause );

orderByClause: ORDERBY (identifier | xpathClause)
(ASC|DESC)?;

returnClause: RETURN processingExpression

processingExpression: identifier
                | xmlClause
                | xpathClause
                | wrapResultClause
                | encodedCoverageExpression
                | mixedClause;
```

Figure 1: xWCPS Syntax.

### 3.2.1 For Statement

The "for" statement snippet is: `{for variable_name in for_expression}`.
It can also contain the let clause allowing variable definition that can be used later on. The for clause binds a variable to each item returned by the in expression. There are 3 options that can be used in a 'for' statement:

- Use all available coverages: *
- Use all coverages of a specific service: *@endpoint (endpoint can be a url with double quotes or an alias)
- Use specific coverages: coverageId or coverageId@endpoint (endpoint can be a url with double quotes or an alias)

### 3.2.2 Let Statement

The let statement snippet is:

```
{let variable_name := wcps_clause;}
```

The let clause can initialize variables following an assignment expression that finishes with a semicolon. The use of the let clause can greatly reduce repetitiveness, making xWCPS extremely less verbose than WCPS. Moreover, arithmetic operations can be executed between defined variables.

### 3.2.3 Where Statement

The where statement is used to specify one or more metadata or coverage related criteria for filtering down the returned result. Currently combined data and metadata join operations are not allowed in the context of xWCPS. Every XPath or WCPS expression evaluating to a boolean result is a valid xWCPS comparison expression. To declare an xPath expression the "::" notation should follow the variable. That notation fetches the metadata of the coverage where the xPath is evaluated.

### 3.2.4 Order by Statement

The Order by statement has the following syntax:

```
{order_by_expression (asc | desc)}
```

Results can be sorted using ORDER BY. Like in FLWOR expressions, the construct takes one or more order expressions that each can have an optional order modifier (ASC or DESC).

The order by clause is used to rank the returned coverages based on an XPath clause applicable on their metadata. If direction is not defined explicitly, ascending is used by default.

### 3.2.5 Return Statement

The return statement of a query specifies what is to be returned and the way that this result should be represented. It can contain textual results, structured XML results, WCPS encoded (i.e. png, tiff, csv) results or combinations of binary and textual data as mixed results. xWCPS acts as a wrapper construct on top of XPath 2.0 and WCPS, thus it doesn't offer any language specific operations. Thus we can have the following options:

- Use the encode function of WCPS -> WCPS result
- Use "::" operator -> Fetch metadata -> XML result
- Use an xPath 2.0 expression / function -> XML result
- Use the new "mixed" function to combine both -> Multipart result

## 3.3 Use Cases

The features and functionality introduced with xWCPS are presented in this section through a number of use cases, examples. The queries represent the expressive power of our language and its superiority over WCPS in array database search. In the context of the EarthServer project we have tested the effectiveness of xWCPS by searching over array databases with terabyte of data and metadata by registering the services and their metadata into the catalogue. Six services are part of the EarthServer project and all of them are making available terabytes of data. More information is available in the public reports of the project.

### 3.3.1 Retrieving Data and Metadata using Special Characters

XQuery was a key feature of xWCPS 1.0. Now in its 2nd revision, xWCPS is based on XPath in order to accomplish user friendliness and simplified queries to retrieve data and metadata. Special characters are introduced for expressiveness in order to easily retrieve all coverages and/or filter them by endpoints. The example below shows a query that uses both * and @ special characters to fetch all coverages from a specific service endpoint and return part of the actual coverage as a result. The encode function of WCPS defines the returned result in this specific case.

```
{for $c in *@ECMWF
return encode($c[ansi("2001-07-
31T23:59:00")] * 1000 , "png")}
```

while the following one shows a query that fetches the metadata of a specific coverage using **::** special character.

```
{for $c in precipitation@ECMWF
return $c::}
```

### 3.3.2 Building Coverage Filtering Queries using XPath

Filtering metadata of a coverage through XPath can be applied in both where and return clause. In where clause to decrease the number of results and in return clause to manipulate what is presented as a result. The following example has is accommodating both filter options by filtering an XML attribute for a specific value and then setting that attribute as the result. In this example, the result contains only XML metadata.

```
{for $c in *@ECMWF
where $c:://RectifiedGrid[@dimension=2]
return $c:://RectifiedGrid}
```

### 3.3.3 Building Coverage Ordering Queries using XPath and Let Clause

xWCPS supports the 'let' clause, which allows assigning complex expressions to variables and re-using them for subsequent references, avoiding repetitiveness. In the following example a variable called '$orderByClause' is assigned with the id of every coverage that matches the 'for' clause. This variable is firstly used to order the results and then to be presented to the user as the returned value. Let clause holds the result of a metadata expression filtered by XPath.

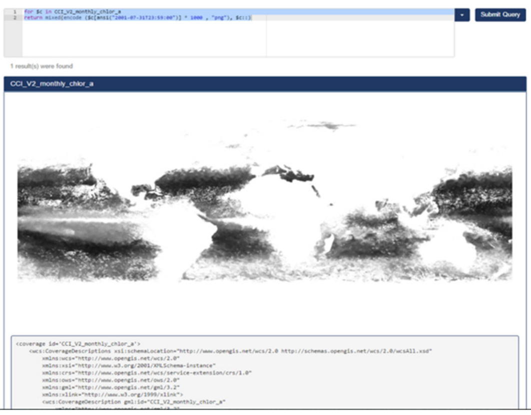
```
{for $c in *@ECMWF
let $orderByClause :=
$c::/wcs:CoverageId/text();
orderby $orderByClause desc
return $orderByClause}
```

### 3.3.4 Retrieving a Mixed Form Containing Data and Respective Metadata

An important feature of xWCPS is the ability to return the data accompanied with their metadata reducing the amount of queries required before and allowing the user to retrieve only one result containing both. This can be achieved using the 'mixed' clause of xWCPS as can be seen in the example below:

```
{for $c in CCI_V2_monthly_chlor_a
return mixed(encode ($c[ansi("2001-07-
31T23:59:00")] * 1000 , "png"), $c::)}
```

In the query above, the usage of the mixed clause will return a result that contains the actual coverage processed as the encode function defines together with the full set of metadata that accompanies it.



```
for $c in CCI_V2_monthly_chlor_a
return mixed(encode ($c[ansi("2001-07-31T23:59:00")] * 1000 , "png"), $c::)
```

1 result(s) were found

CCI_V2_monthly_chlor_a

```
<coverage id='CCI_V2_monthly_chlor_a'>
    <wcs:CoverageDescriptions xsi:schemaLocation="http://www.opengis.net/wcs/2.0 http://schemas.opengis.net/wcs/2.0/wcsAll.xsd"
        xmlns:wcs="http://www.opengis.net/wcs/2.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:crs="http://www.opengis.net/wcs/service-extension/crs/1.0"
        xmlns:ows="http://www.opengis.net/ows/2.0"
        xmlns:gml="http://www.opengis.net/gml/3.2"
        xmlns:xlink="http://www.w3.org/1999/xlink">
        <wcs:CoverageDescription gml:id="CCI_V2_monthly_chlor_a"
```

Figure 2: xWCPS Web Application.

The source result of an xWCPS query is in JSON format and it contains both the metadata and the actual coverage in base64 format. For simplicity the xWCPS web application supports the execution of xWCPS queries, including all the above examples. Figure 2 shows how a mixed result is displayed in the web application.

## 4 IMPLEMENTATION

### 4.1 Base Architecture

xWCPS constitutes of two distinct implementations. Initially, a query parser has been implemented to support the query translation based on the language definition presented before. It uses the ANTLR 4 framework (ANTLR, 2018) and it translates the xWCPS queries to source code. The language is specified using a context-free grammar, which is expressed using Extended BackusNaur Form (EBNF). Open source grammars for WCPS and XPath are extracted from (ANTLR WCPS, 2018) and (ANTLR XPath, 2018) respectively.

The xWCPS engine implementation exploits FeMME metadata management engine for the metadata query support and it utilizes registered Rasdaman servers for processing the (geospatial) array queries following the WCPS syntax.

The overall architecture of the system is shown in Figure 3.

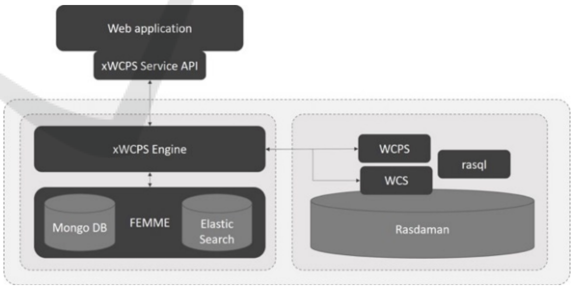


Figure 3: xWCPS Engine Architecture.

xWCPS offers either a web application for end users or a REST API for machine to machine interaction. Any valid xWCPS query can be executed and the results are returned to the consumer. The flow for executing a query is the following: Initially the for and where clauses of the query are analysed, producing a composite query which is evaluated against FeMME. Following the composite query strategy, XPath evaluation on FeMME can be optimized by restricting the number of coverages that are considered for the XPath execution.

SCIENCE AND TECHNOLOGY PUBLICATIONS

As soon as the first evaluation step is completed the return statement is executed using the items returned in the first step. Depending on its contents, return can utilize either FeMME or Rasdaman. Encode function is evaluated in its entirety using Rasdaman and its supported geospatial operations. Other available expressions, like XPath and metadata retrieval, use FeMME to generate the returned result.

## 4.2 FeMME: Metadata Management Engine

Metadata play a significant role in the evaluation of an xWCPS query. It is the means of identifying and filtering the available coverages through the executed queries. The goal of the metadata management engine is to amalgamate all this information into one catalogue offering federated metadata search upon coverages' descriptive information. In order to support the execution of the metadata part of the queries such an engine, termed "FeMME", Federated Metadata Management Engine, has been designed and implemented. The main principles it adheres to, are the metadata schema agnosticism, in order to support storage, querying and manipulation of descriptive metadata from various data sources and querying (XPath) performance efficiency.

FeMME has been designed aiming on being pluggable and supporting the storage of metadata available through different protocols and standards. To this end, a number of sub-components enable the harvesting of metadata for every available collection of coverages, which are first initialized with the WCS available metadata and can then be enriched from other catalogues supporting them.

xWCPS uses FeMME as its central point for identifying and retrieving the required information for each collection of coverages and for gaining access to the Describe Coverage metadata and executing the XPath queries.

### 4.2.1 XPath Performance Efficiency

In order to overcome the inherent memory and speed limitations of in memory XPath, as proved to be the main limitation of the initial implementation, it was decided to utilize the speed and flexibility of NoSQL systems to implement XPath.

The technology used initially was MongoDB. The approach followed at first was to flatten an XML document and store each XML element as a separate document in MongoDB. Building a custom parser allowed us to transform an XPath to a MongoDB query and evaluate it in the database. An unforeseen issue was that this method of "indexing" an XML document resulted in the creation of a large number of documents. For example, for a typical response the number of documents produced was over 1000. As the number of indexed metadata increased, so did the overhead.

As a result, a different approach was followed. Each XML document is transformed to one JSON object, reflecting the XML document's structure and hierarchy. Each XML element would map to a JSON node and children elements to children nodes. Namespaces and attributes would be transformed to children nodes. This way it is possible to transform XML to JSON and vice versa without losing any information.

In order to achieve better performance different technologies were evaluated. ElasticSearch was chosen to provide the storing and querying capabilities. ElasticSearch also stores data as JSON documents but, in contrast to MongoDB, indexes every field of a JSON document. This fact promised much better performance for queries that could query for a value at any level of the document.

## 4.3 Federated Geospatial Queries Execution

The execution of the geospatial, array queries of xWCPS are executed remotely, by interacting with the appropriate array database engine (i.e. rasdaman) using a WCPS query. FeMME holds all the required information for each registered array database addressed by xWCPS in order to identify the appropriate service endpoint that holds the coverages defined, or is specified in the xWCPS query.

The system can interact with more than one data management service at one query, allowing the concurrent retrieval of data that are not part of the same engine. This feature is considered to vastly simplify the implementation of applications for array data aggregation and presentation from multiple sources through a unified way, rendering end-user application development quite straight forward.

The performance of this part of the execution is highly relying on the interconnection and performance of the array database engines that comprised the implied federation. It has to be noted that the volume of the data transported may become a reason for bottlenecks and delays.

## 5 APPLICATIONS

One of the main applications is applying the solution over the Meteorological Archival and Retrieval System (MARS). MARS is the main repository of meteorological data at the European Centre for Medium-Range Weather Forecasts (ECMWF). MARS hosts operational and research data, as well as data from special projects. The archive holds Petabytes of data, mainly using GRIB format for Meteorological fields and BUFR format for Meteorological Observations. Most of the data produced at ECMWF daily is archived in MARS, and therefore available to users via its services.

The MARS archive integration aims to bring the more than 100PB MARS archive of ECMWF to its audience via the WCS and WCPS standards. Due to its enormous size it is practically infeasible to ingest the data into a system capable of exposing those via the aforementioned standards, as this would require twice the storage space. Thus, a different approach had to be designed to allow only the required subset of data that are addressed by a WCS/WCPS operation to moved out of the archive when required.

xWCPS was chosen as the best candidate to address these requirements due to its simplicity, expressiveness and filtering capabilities. As the project lays its array processing capabilities on the Rasdaman engine, the objective is to move as little data as feasible into the Rasdaman data store, on demand and offload the remaining processing to the Rasdaman engine. Utilizing the metadata management capabilities of FeMME allowed the on the fly data retrieval from MARS and subsequent ingestion in Rasdaman. As soon as the MARS data lie in Rasdaman, the aforementioned workflow of an xWCPS query can be carried out.

A descriptive diagram of how the MARS system works, is shown in Figure 4.

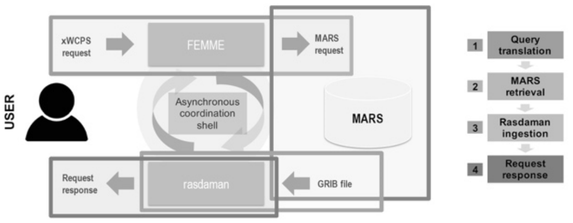


Figure 4: MARS System Integration.

FeMME and xWCPS subsystems are surrounded by a web UI that allows the exploration of coverages under a familiar virtual globe, offered by NASA Web World Wind 3D virtual globe technology [NASA WWW] that allows rendering the coverage data and metadata (bounding boxes). The combined platform allows handling, retrieval processing and visualization of various Coordinate Reference Systems, making it possible to utilize the same stack for rendering Earth as well as other solar body data.

## 6 EVALUATION

We evaluate our approach from two perspectives, (a) the language definition and (b) its prototypical implementation.

Regarding the language definition, as opposed to xWCPS v1.0, we have an apparently simpler grammar, giving away quite a significant part of XQuery processing features, which is the main drawback of the initial approach. However, this move in, in par with the OGC-Extended Coverage Implementation Schema which also suggests XPath for querying hierarchical metadata. It has to be noted that hierarchical metadata are assumed to be the prominent model, be they in json or xml form, although not the only or most powerful model in place. Naturally due to the specification size, potential clashes among WCPS and XPath are quite fewer and as such are resolved in a more intuitive manner, while the syntactic sugar added supports common use cases identified within the hosting project course and based on the feedback provided by the rest of the partners, experts in the geospatial domain.

xWCPS eases significantly implementation of applications, as it removes the need for multistep approach followed by coverage data consumers, which encloses at least the steps of locating the dataset, extracting the significant metadata for processing it and finally processing its content. This approach not only implies round trips but also requires that the client understands the form of the data/metadata infrastructure and reduces the opportunity of server-side optimisation of data management and processing.

Regarding the aspect of prototyping, the current implementation is based on Rasdaman array database for WCPS queries, on one hand and FeMMe engine on the other for metadata retrieval and the two parts of execution are orchestrated by the prototype's execution engine. This has the drawback that little optimisation can be performed at query time as internal engine structures and processes run separately. Nevertheless, the built-in heuristics (e.g. assume that XPath execution is always faster than accessing the array data) manage to avoid common pitfalls. In return, for this approach, we achieve a much cleaner implementation that does not reside on a particular engine's characteristics and may be moved from one context to another (e.g. a different array DB or metadata management engine), which is a quite stronger requirement at the stage of prototyping a language implementation.

## 7 FUTURE WORK

Having provided an implementation of xWCPS and proved its potential and usefulness, there is still space for several improvements both in the language definition and in the implementation of the engine that supports it. Full support of the XPath 2.0 specification is the first priority to work on allowing more efficient filtering of the available metadata leading to queries that require less transfer of data and increasing the response time. Subsequently, we plan to work on optimizing the response time of the xWCPS queries, by improving the FeMME metadata engine which is the core component of the engine, working on the transformation of a single component to a distributed one. Apart from that, user experience like auto-complete and a visual query editor could be further investigated based on the clients' feedback together with security aspects arising between the different layers of the architecture and the number of Rasdaman engines registered in the catalogue.

## ACKNOWLEDGEMENTS

This work has been partially supported by the European Commission under grant agreement H2020 654367, "Agile Analytics on Big Data Cubes (EarthServer 2)" and is powered by rasdaman array database engine and NASA Web World Wind 3D virtual globe.

## REFERENCES

Liakos, P., Koltsida, P., Kakaletris, G., and Baumann, P. (2015). xWCPS: Bridging the gap between array and semi-structured data. In *Knowledge Engineering and Knowledge Management, pages 120–123. Springer*.

Baumann P., (2012). OGCr WCS 2.0 Interface Standard — Core. OGC 09-110r4, version 2.0. OGC.

Baumann, P. (2010). The OGC web coverage processing service (WCPS) standard. *GeoInformatica, 14(4):447–479.*

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., and Widmann, N. (1998). The multidimensional database system rasdaman. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 575–577. ACM.*

Brown, P. G. (2010). Overview of SciDB: Large scale array storage, processing and analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 963–968. ACM.*

Baumann, P., Mazzetti, P., Ungar, J., Barbera, R., Barboni, D., Beccati, A., Bigagli, L., Boldrini, E., Bruno, R., Calanducci, A., Campalani, P., Clements, O., Dumitru, A., Grant, M., Herzig, P., Kakaletris, G., Laxton, J., Koltsida, P., Lipskoch, K., Mahdiraji, A.R., Mantovani, S., Merticariu, V., Messina, A., Misev, D., Natali, S., Nativi, S., Oosthoek, J., Pappalardo, M., Passmore, J., Rossi, A.P., Rundo, F., Sen, M., Sorbera, V., Sullivan, D., Torrisi, M., Trovato, L., Veratelli, M.G., Wagner, S., 2016. Big data analytics for earth sciences: the EarthServer approach. *Int. J. Digital Earth 9:3–29.*

W3c.org. (2018). *XML Path Language (XPath) 2.0 (Second Edition)*. [online] Available at: http://www.w3c.org/TR/xpath20 [Accessed 2 Jan. 2018].

OGC. (2017). The OpenGIS® Abstract Specification Topic 6: Schema for coverage geometry and functions, Version 7. [online] Available at: http://portal.opengeospatial.org/files/?artifact_id=19820 [Accessed Dec. 2017].

ANTLR. (2018). ANTLR. [online] Available at: http://www.antlr.org [Accessed 5 Dec. 2017].

Earthserver.eu. (2018). *Home | EarthServer.eu*. [online] Available at: http://earthserver.eu [Accessed 2 Jan. 2018].

ANTLR WCPS. (2018). WCPS Grammar. [online] Available at: http://www.rasdaman.org/browser/applications/petascope/petascope_main/src/main/java/petascope/wcps/parser/wcps.g4 [Accessed 1 Jan. 2018].

ANTLR XPath. (2018). antlr/grammars-v4. [online] Available at: https://github.com/antlr/grammars-v4/tree/master/xpath [Accessed 1 Jan. 2018].

NASA WWW. (2018). Web World Wind. Available at https://worldwind.arc.nasa.gov/web/ [Accessed 1 Jan. 2018].