D3.10 Big Data Toolbox II

WP3 – Large Scale Demonstrators

Authors: Dimitar Misev, Peter Baumann, Gedas Vaitkus

Date: 28.07.20

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824478.

Full Title	**Promoting the international competitiveness of European Remote Sensing companies through cross-cluster collaboration**		
Grant Agreement No	824478	**Acronym**	PARSEC
Start date	1st May 2019	**Duration**	30 months
EU Project Officer	Milena Stoyanova		
Project Coordinator	Emmanuel Pajot (EARSC)		
Date of Delivery	**Contractual** 31.07.2020	**Actual**	28.07.2020
Nature	Other	**Dissemination Level**	Public
Lead Beneficiary	RASDAMAN		
Lead Author	Dimitar Misev	**Email**	misev@rasdaman.com
Other authors	Peter Baumann (RASDAMAN), Gedas Vaitkus (GEOMATRIX)		
Reviewer(s)	Ola Majorczyk (EVERSIS)		
Keywords	big data, EO, datacubes, array databases, OGC services, rasdaman, sagris		

Document History

Version	Issue date	Stage	Changes	Contributor
1.0	29.06.2020	Draft	First draft	RASDAMAN
2.0	22.07.2020	Draft	Contribution from GMX	GMX, RASDAMAN
2.1	24.07.2020	Draft	Ready for review	RASDAMAN
2.2	28.07.2020	Final	Integrate review comments	RASDAMAN, EVERSIS

Table of Contents

Table of Figures

Table of Tables

List of Acronyms

API	Application Programming Interface
CRS	Coordinate Reference System
EO	Earth Observation
OGC	Open Geospatial Consortium
S3	Object storage specification by Amazon Web Services
WMS	Web Map Service
WCPS	Web Coverage Processing Service
WCS	Web Coverage Service

Executive Summary

This report documents the PARSEC Big Data Toolbox for management and access of spatio-temporal data, based on the rasdaman datacube engine and the SAGRIS data pre-processing suite. The Toolbox offers access to large EO data (primarily Sentinel-1 and 2, plus further types of data as needed) via standard OGC datacube API, namely WCS, WCPS, and WMS.

First the overall platform design and architecture of the Big Data Toolbox are presented. This is followed by comprehensive sections on rasdaman and SAGRIS, covering details from architecture and main features, to software / hardware requirements for deployment and administration instructions. Finally, the first prototype of the Big Data Toolbox is briefly showcased.

1 Introduction

The PARSEC Big Data Toolbox consists of many components working together to offer flexible and fast access to analysis-ready EO datacubes. The primary technologies behind the scene are *rasdaman*, a pioneering engine for scalable datacube management, and *SAGRIS*, a powerful EO data pre-processing toolbox primarily applied to Sentinel data products.

Data on local or network filesystem, including pre-processing results from SAGRIS, is ingested or registered as datacubes in rasdaman via WCS-T. The datacubes are subsequently made available via standard OGC interfaces: WCS, WCPS, and WMS (Figure 1).

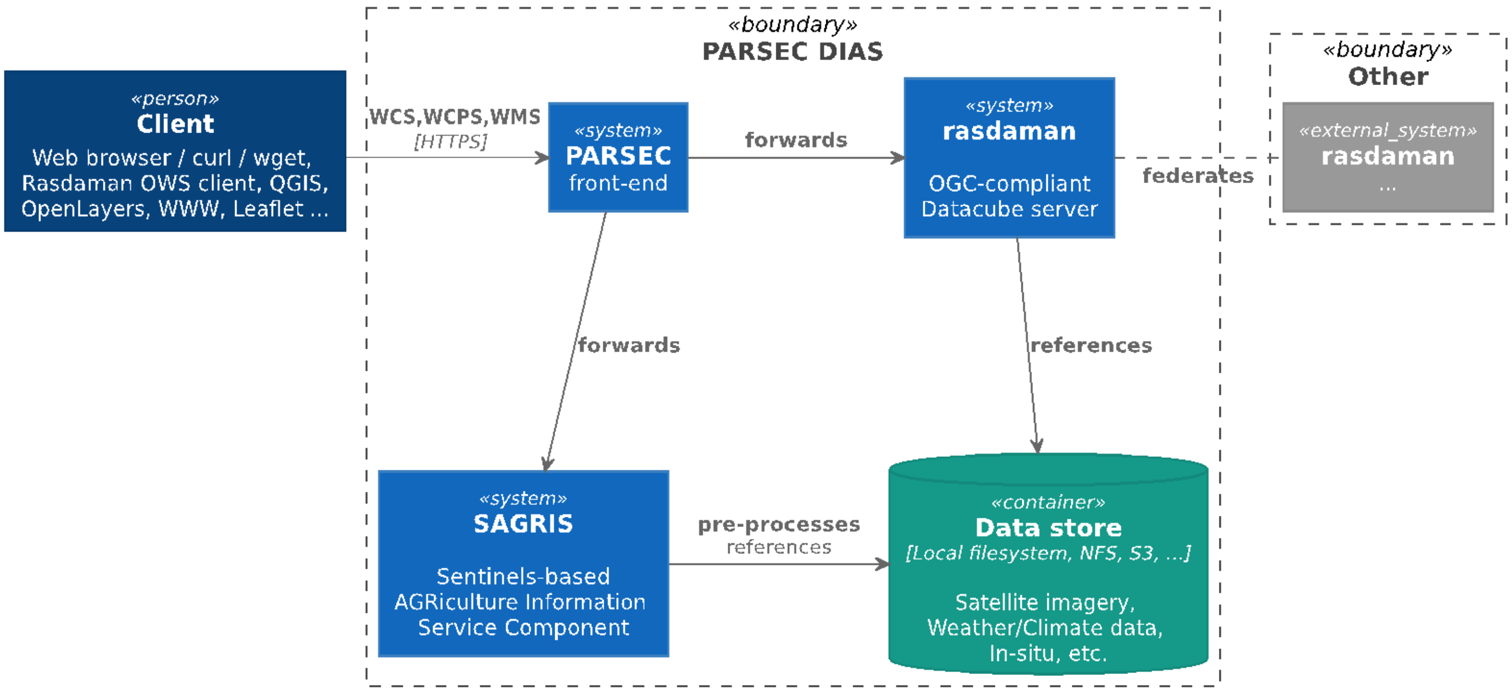

«boundary»
PARSEC DIAS
«person»
Client
Web browser / curl / wget, Rasdaman OWS client, QGIS, OpenLayers, WWW, Leaflet ...
WCS,WCPS,WMS
[HTTPS]
«system»
PARSEC
front-end
forwards
«system»
rasdaman
OGC-compliant Datacube server
federates
«boundary»
Other
«external_system»
rasdaman
...
forwards
references
«system»
SAGRIS
Sentinels-based AGRiculture Information Service Component
pre-processes
references
«container»
Data store
[Local filesystem, NFS, S3, ...]
Satellite imagery, Weather/Climate data, In-situ, etc.

Figure 1 High-level overview of the PARSEC Big Data Toolbox.

SAGRIS and rasdaman are deployed on different DIAS platforms, so that satellite image and external data pre-processing is done separately in order to increase service scalability and reduce processing loads on the main PARSEC service container (Figure 2).

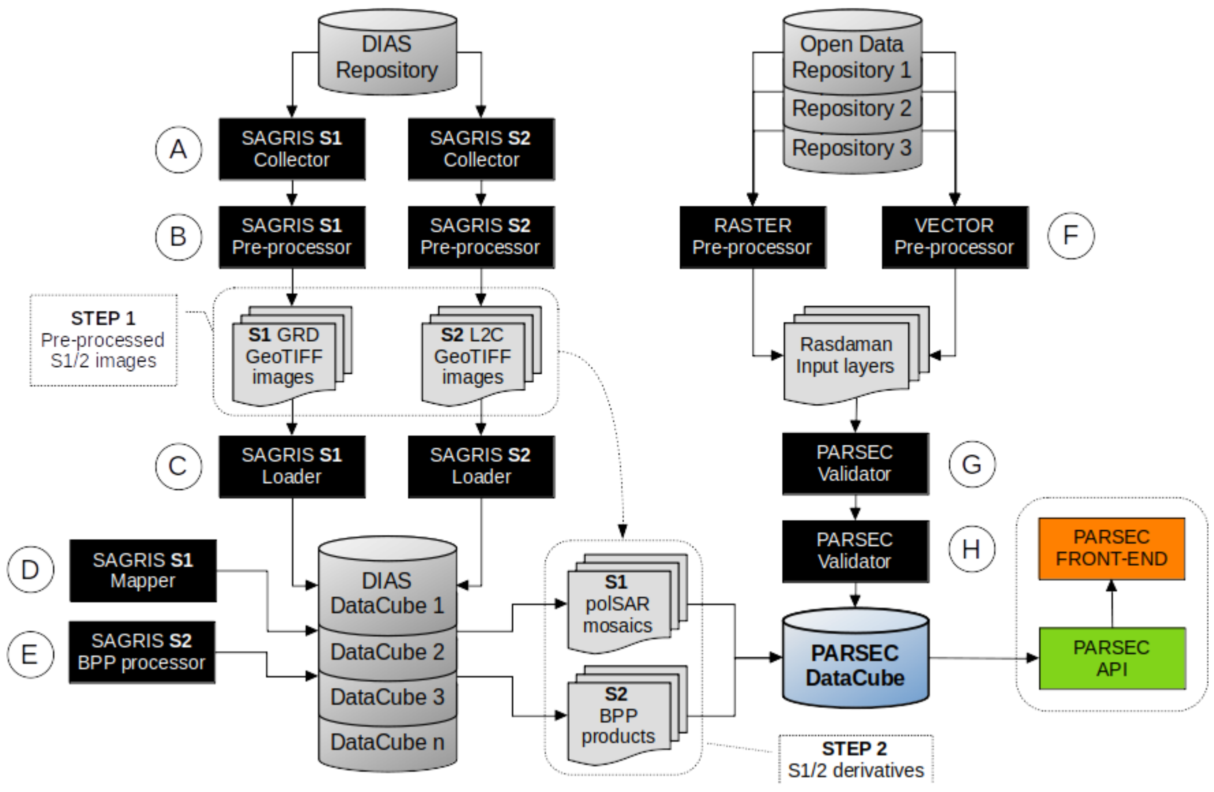


Figure 2 More detailed overview of the integration of SAGRIS pre-processors, external datasets, and PARSEC platform, powered by rasdaman.

The following sections dive deeper into the architecture of rasdaman and SAGRIS and showcase the operational Big Data Toolbox currently deployed on the Mundi DIAS.

2 Rasdaman

The goodies of database technology for building flexible, large-scale data management systems are widely known:

- *information integration* brings better consistency and much easier administration;
- *flexibility* by replacing static APIs by dynamic query languages;
- *scalability* achieved through advanced storage and processing techniques;
- *maturity* of decades of development focused on data integrity and functionality richness.

Unfortunately, these advantages until recently could be reaped only for alphanumeric data types. Support for raster EO data in particular (also known as datacubes, multidimensional arrays, gridded data, sampled data, etc.) has been missing from traditional databases.

This gap is closed by the rasdaman technology which offers distinct datacube management features. Its conceptual model supports arrays of any number of dimensions and over virtually any cell (“pixel”, “voxel”) type. The rasdaman query language, rasql, is crafted along standard SQL and gives high-level, declarative access to any raster data; recently, it has been standardized as ISO SQL/MDA. Its architecture principle of tile stream processing, together with highly effective optimizations, has proven scalable into multi-Petabyte object sizes. Rasdaman has been developed since 1996 and has reached a high level of maturity with operational use since many years.

Technically, rasdaman is a domain-neutral array DBMS, which makes it suitable for application in all domains where multidimensional array data is of relevance. Especially EO spatio-temporal data is well supported out of the box, with compliant interfaces for OGC WCS, WCPS, and WMS.

2.1 Client-server architecture

Rasdaman runs as a *server* that takes care of data management and processing; *clients* connect to via HTTP(S) to submit queries to the server and retrieve processing results (Figure 3). A large variety of clients speak the OGC WCS/WMS language and are hence compatible with the rasdaman server; see the companion D3.2 Big Data Toolbox Training Manual I document for a detailed overview.

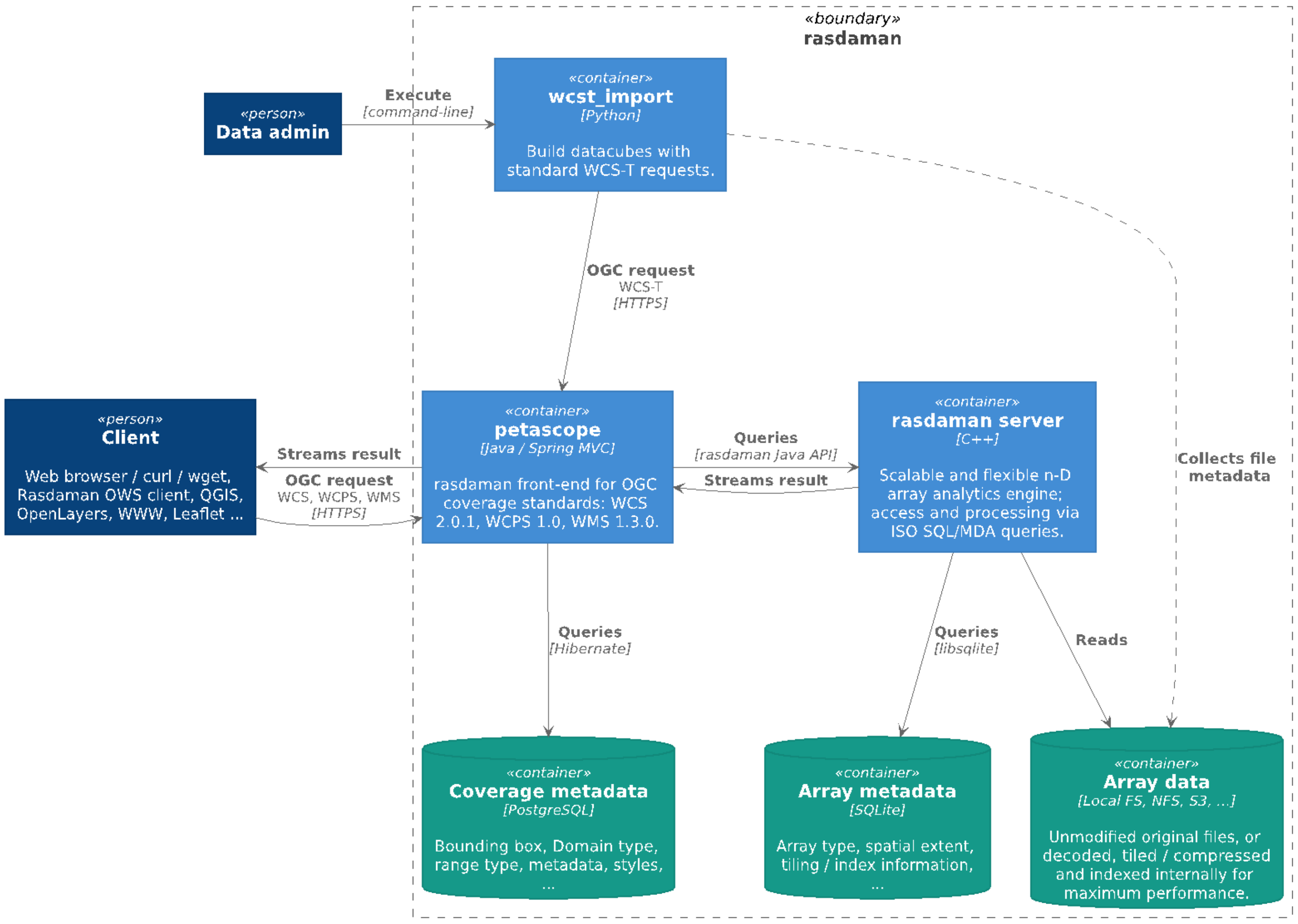

«boundary» rasdaman
«person» Data admin
Execute [command-line]
«container» wcst_import [Python] Build datacubes with standard WCS-T requests.
OGC request WCS-T [HTTPS]
«person» Client Web browser / curl / wget, Rasdaman OWS client, QGIS, OpenLayers, WWW, Leaflet ...
Streams result
OGC request WCS, WCPS, WMS [HTTPS]
«container» petascope [Java / Spring MVC] rasdaman front-end for OGC coverage standards: WCS 2.0.1, WCPS 1.0, WMS 1.3.0.
Queries [rasdaman Java API]
Streams result
«container» rasdaman server [C++] Scalable and flexible n-D array analytics engine; access and processing via ISO SQL/MDA queries.
Collects file metadata
Queries [Hibernate]
Queries [libsqlite]
Reads
«container» Coverage metadata [PostgreSQL] Bounding box, Domain type, range type, metadata, styles, ...
«container» Array metadata [SQLite] Array type, spatial extent, tiling / index information, ...
«container» Array data [Local FS, NFS, S3, ...] Unmodified original files, or decoded, tiled / compressed and indexed internally for maximum performance.

Figure 3 Rasdaman client-server architecture.

The *data administrator* is responsible for registering data into *rasdaman datacubes.* For this purpose, usually the user-friendly wcst_import tool is used. Parameterized with a JSON configuration (ingredient) file, it takes care of generating standard-compliant WCS-T requests to *petascope*, the OGC coverages front-end of rasdaman (Figure 4). Petascope is itself a server, but technically also a client of the rasdaman core server. It translates the geo-rich OGC requests into domain-neutral rasql queries, which are then evaluated by rasdaman.

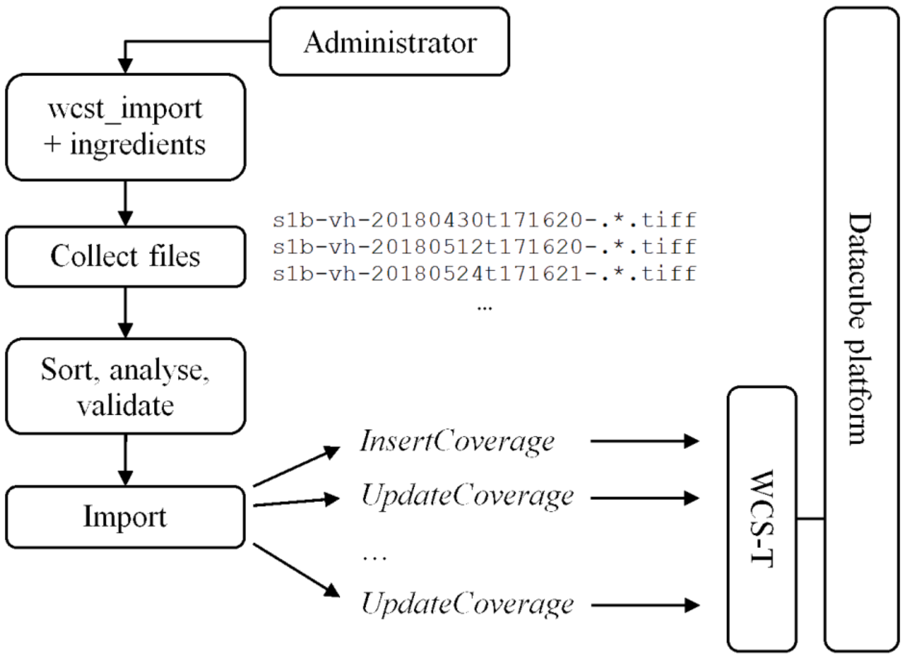


Figure 4 Ingestion process with wcst_import.sh.

Figure 5 shows a detailed architecture diagram of all client and server components; it is especially relevant to server administrators and client developers interacting with the rasdaman client API.

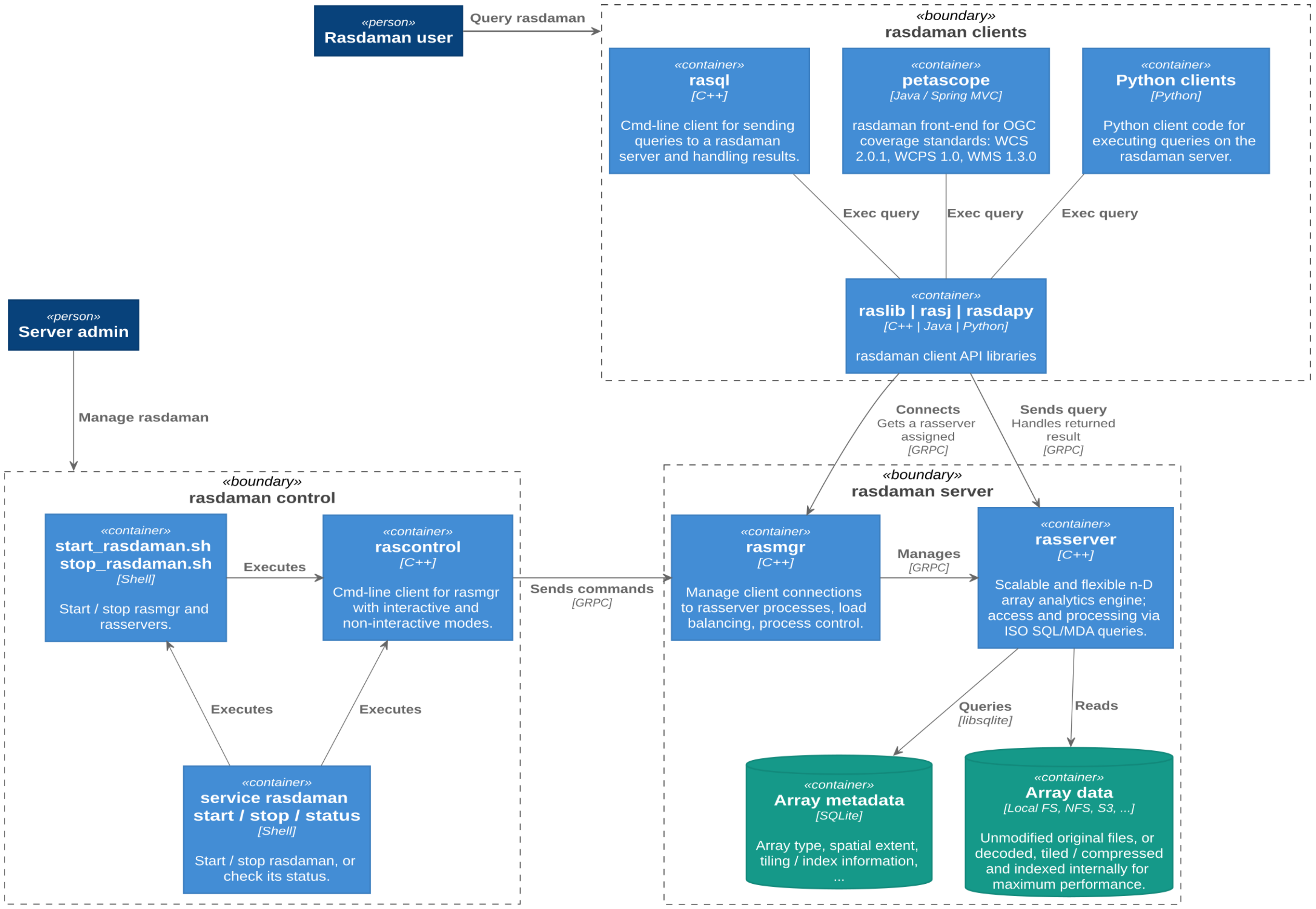

«person» Rasdaman user
Query rasdaman
«boundary» rasdaman clients
«container» rasql [C++] Cmd-line client for sending queries to a rasdaman server and handling results.
«container» petascope [Java / Spring MVC] rasdaman front-end for OGC coverage standards: WCS 2.0.1, WCPS 1.0, WMS 1.3.0
«container» Python clients [Python] Python client code for executing queries on the rasdaman server.
Exec query
Exec query
Exec query
«container» raslib | rasj | rasdapy [C++ | Java | Python] rasdaman client API libraries
«person» Server admin
Manage rasdaman
Connects Gets a rasserver assigned [GRPC]
Sends query Handles returned result [GRPC]
«boundary» rasdaman control
«container» start_rasdaman.sh stop_rasdaman.sh [Shell] Start / stop rasmgr and rasservers.
Executes
«container» rascontrol [C++] Cmd-line client for rasmgr with interactive and non-interactive modes.
Sends commands [GRPC]
Executes
Executes
«container» service rasdaman start / stop / status [Shell] Start / stop rasdaman, or check its status.
«boundary» rasdaman server
«container» rasmgr [C++] Manage client connections to rasserver processes, load balancing, process control.
Manages [GRPC]
«container» rasserver [C++] Scalable and flexible n-D array analytics engine; access and processing via ISO SQL/MDA queries.
Queries [libsqlite]
Reads
«container» Array metadata [SQLite] Array type, spatial extent, tiling / index information, ...
«container» Array data [Local FS, NFS, S3, ...] Unmodified original files, or decoded, tiled / compressed and indexed internally for maximum performance.

Figure 5 Detailed architecture diagram of rasdaman.

2.2 Software requirements

Rasdaman can be deployed on a variety of Linux distributions. Tested and supported with standard DEB and RPM packages are *Ubuntu 16.04 / 18.04 / 20.04*, and *CentOS 7*.

Dependencies

During installation, the packages automatically pull required dependencies to be installed as well:

General libraries

- *libsqlite*, *libsqlite-dev*, *sqlite3* – required for storing arrays in a filesystem directory and the rasdaman technical metadata in SQLite;
- *libssl-dev*, *libncurses5-dev*, *libedit-dev*, *libboost-dev* (v1.48+), *libffi-dev* – required for various system tasks.

Data encoding/decoding

- *libgdal-dev* – required for data format support (TIFF, JPEG, PNG, etc.);
- *libhdf4g-dev* – required for HDF4 support;
- *libnetcdf-dev*, *python3-netcdf4* – required for NetCDF support;
- *libeccodes0*, libeccodes-dev, *python3-grib* - for GRIB data support;
- *libtiff-dev*, *libjpeg-dev*, *ligpng-dev* - internal encoder/decoder implementations for TIFF, JPEG, or PNG formants.

Geo data support

- Tomcat (or another suitable servlet container) – required for running the petascope and SECORE Java web applications, unless they are configured to start in standalone mode;
- PostgreSQL – required for running petascope, unless it's configured to use an embedded database like H2 / HSQLDB
- *python3-dateutil python3-lxml python3-pip python3-gdal python3- netcdf4* (required by wcst_import, a tool for importing geo-referenced data into rasdaman / petascope).

Installation on-premise

In PARSEC, rasdaman is available as a Big Data Toolbox service for usage by stakeholders. However, installation on-premise is always possible as well if required, e.g. after the project finishes or if deployment of custom / proprietary data is necessary. Both the open-source rasdaman community and the more scalable and feature-rich rasdaman enterprise are straightforward for deployment on the supported platforms; on request, support for further platforms could be considered.

Network configuration

For deployment, rasdaman will require some ports to be open on the machine. At a minimum, the Tomcat service with OGC interfaces needs to be exposed publicly; by default this is port 8080, however, it is recommended to use port 80 instead if possible as on many networks only this port will be open. If rasdaman is to be connected into a federation, then furthermore ports 7000 and 7001 need to be open, as well as 7002-7011 (according to the configuration in rasmgr.conf), so that communication with other rasdaman servers in the federation is possible.

2.3 Hardware requirements

Rasdaman is not resource-intensive on its own, and hardware requirements are guided mainly by data volume. For a large public service used by multiple clients, the following recommendation can be made:

Resource type	Recommendation
CPU	`Min. 8-core`
Main memory (RAM)	`Min. 64 GB`
Local disk	`Min. 1 TB`

Table 1 Hardware requirements.

2.4 Main features

Datacube building

Building raster datacubes of any dimension is supported; usually this means time-series, occasionally 2D mosaics. "Building" a datacube means one of

- *data ingestion*: decode, partition, compress and index data internally = slower building process, very fast querying
- *data registration*: index only links to the data = very fast building process (<1 second per scene), but slightly slower querying as data is decoded on-the-fly

Most common raster data formats are supported through integration with GDAL, along with native NetCDF, HDF, and GRIB converters.

Pre-processing data while building a datacube is supported with scene-level hooks, which allow executing shell commands before/after a scene is imported in rasdaman, e.g., reprojecting, format conversion, removing temporary files, etc.

Datacube processing and analytics

Rasdaman implements a flexible and powerful query language based on Array Algebra, recently standardized as ISO SQL/MDA; the equivalent OGC standard is WCPS. When needed, rasdaman can interface to external libraries via UDFs (User-defined functions).

Many standard tasks, like band ratios for various indices, applying masks, etc. are extremely optimized and almost as fast to compose on-the-fly as reading precomputed results would be. The same holds for aggregating data and typical time-series analysis. Query processing is parallelized over the available cores, and federating rasdaman instances provides the opportunity for scaling beyond a single machine.

Support for fine-grain access control based on triggers, along with limits on how much data can be accessed or transferred.

2.5 Operating instructions

Once rasdaman has been installed, a rasdaman service script allows starting/stopping or checking the status:

```
$ service rasdaman start
$ service rasdaman stop
$ service rasdaman status
```

Similarly, the tomcat and postgresql services can be started and stopped. Note that rasdaman runs with non-root user rasdaman. By default, rasdaman is installed in /opt/rasdaman with the following directory structure:

Directory	Description
bin	rasdaman executables, e.g. rasql, start_rasdaman.sh, …
data	Path where the server stores array tiles as files; this directory can get big, so it is recommended to make it a link to a sufficiently large disk partition.
etc	Configuration files, e.g. rasmgr.conf
Include	C++ API development headers.
lib	C++ and Java API libraries.
log	`rasmgr` and `rasserver` log files.
share	Various artefacts like documentation, python/javascript clients, example data, migration scripts, etc.

Table 2 Installation directory structure.

The table below lists the rasdaman executables and scripts found in the bin directory.

Directory	Description
rasserver	Client queries are evaluated by a `rasserver` worker process.
rasmgr	A manager process that controls `rasserver` processes and client/server pairing.
rascontrol	A command-line frontend for `rasmgr`.
rasql	A command-line client for sending queries to a `rasserver` (as assigned by the `rasmgr`).
wcst_import.sh	Tool for convenient and flexible ingestion of geo-referenced data into petascope.

Table 3 rasdaman executables.

Server rasdaman configuration files can be found in the etc directory. The configurations are automatically loaded upon rasdaman start. After any modification, a restart must be performed for the change to take effect.

Directory	Description
rasmgr.conf	allows fine-tunning the rasdaman servers, e.g. number of servers, names, database connection, etc.
petascope.properties	set petascopehttp://rasdaman.org/wiki/PetascopeUserGuide properties, e.g. database/rasdaman connection details, CRS resolver URLs, various feature options
secore.properties	secore (CRS resolver) configuration
log-rasmgr.conf	configure log output of rasmgr
log-server.conf	configure log output of rasservers
log-client.conf	configure log output of clients (e.g. rasql)

Table 4 rasdaman configuration files.

The rasdaman log files are found in the log directory. The server components feed the following files where *uid* represents a unique identifier of the process, and *pid* is a Linux process identifier:

- *rasserver.<uid>.<pid>.log:* rasserver worker logs: at any time several rasservers are running (depending on the settings in rasmgr.conf) and each has a unique log file.
- *rasmgr.<pid>.log:* rasmgr log: there is only one rasmgr process running at any time.
- *petascope.log, secore.log*: petascope and secore log files, if the paths are correctly configured in petascope.properties and secore.properties.

3 SAGRIS

Extraction of business intelligence information on the lowest level of sampling units (agriculture parcels, forestry blocks, etc.) from standard Sentinel-1 and Sentinel-2 products demands high positional accuracy and low distortion of the satellite signal, which is only possible with precise calibration and ortho-rectification of satellite imagery. On the other hand, those are the most processing-intense remote sensing operations – therefore, we designed a series of binary processors for a cloud-based production environment, which enables scaling of parallel processing tasks and using local repositories of satellite images.

Careful pre-processing of Sentinel-1 and 2 products ensures spatio-temporal consistency and thematic accuracy of sampled business intelligence data; therefore, we primarily focus on the usability of satellite data products delivered by the SAGRIS workflow, offering a series of practical and efficient solutions:

- Instead of using generic Range Doppler Terrain Correction and despeckling of Sentinel-1 products, we perform full-scale terrain flattening and ortho-rectification with high-resolution DEM and preserve the original polSAR back-scatter values;
- Instead of using a collection of partially overlapping Sentinel-2 granules, we transform granules into a single projection, reconstruct seamless single-pass scenes, bind them into a consistent time-series and only then perform sampling of per-parcel spectral information;
- Instead of using the default 64-bit data depth for Sentinel-1 data products, we multiply by 1000 and convert to 16-bit integers, thus reducing the size of pre-processed imagery by 5 times and speeding up the further processing operations, which have to deal with integer values instead of floats.

3.1 Binary pre-processors

For PARSEC data pre-processing, we propose 3 binary SAGRIS processors, designed for pre-processing of standard Sentinel-1 and 2 products provided by ESA in Standard Archive Format for Europe (SAFE[1]) format into production-ready standard GeoTIFF[2] data format Sentinel-1 and Sentinel-2 products:

1. Sentinel-1 pre-processor
2. Sentinel-2 pre-processor
3. Sentinel-2 image aggregator

All SAGRIS processors are designed as “headless” stand-alone programs to be executed as separate commands or in an iterative batch mode combined with input/output parameters.

Sentinel-1 and 2 pre-processors have built-in capabilities of iterative scanning of input folders and detecting newly uploaded input files, as well as iterative scanning of the output folder and detection

[1] https://www.eumetsat.int/website/home/Data/Products/Formats/SentinelFormats/index.html
[2] http://www.gdal.org/frmt_gtiff.html

of the newly dumped pre-processed images, which enables operational flexibility of parallel implementation of multiple dynamic “sub-processes” sharing the same processing batch.

Sentinel-2 pre-processor has a built-in capability of detection standard processing levels (L1C and L2A) provided by ESA; therefore, it will skip the time-consuming Atmospheric Correction phase of L1C product in case if corresponding L2A product is already available in the same input folder (per sub-process). In the same way, to save processing time, the Sentinel-1 pre-processor will skip the most time-consuming SNAP processes (calibration, terrain flattening, ortho-rectification) in case if a pre-processed BEAM DIMAP[3] data package is available in the output folder. This allows for a rapid production of large-scale seamless Sentinel-1 data coverages in different plain projections using the same archive of pre-processed Sentinel-1 DIMAP products.

Sentinel-2 image aggregator has a built-in capability of aggregation separate pre-processed granules (Sentinel-2 L2B) into seamless satellite passes (Sentinel-2 L2C).

Both Sentinel-1 and Sentinel-2 pre-processors offer functionality for dividing large regions into production tiles (both Sentinel-1 and 2 processors), which enables scaling the production into continental and even global extents.

3.2 Software requirements

All binary processors are developed on the Linux operating system platform and they require Linux system components to operate properly, so essentially, those binaries are exclusive Linux binary programs. We recommend Ubuntu 16.04 LTS Server[4] 64-bit operating system (stand-alone or Virtual machine) as the production platform. On the operating system level, all binary programs require Python 2.7 platform to be installed on the processing servers or virtual machines.

Sentinel-1 pre-processor is based on standard open-source software components: ESA SNAP[5] version 6+ (recommended version 7.x) and GDAL[6]/OGR[7] version 2.x . ESA SNAP software should include only SNAP[8], S1TBX[9] and S2TBX[10] components. The only SNAP component used by Sentinel-1 binary pre-processor is Graph Processor GPT; therefore, no SNAP GUI is needed for software operation. This suggests that whenever standard “headless” SNAP instances are available, the processors could be deployed there. Due to inefficient use of hardware resources by Java platform powering ESA SNAP software, Sentinel-1 pre-processing workflow had to be split into separate functional steps and hard-coded into the software, therefore, currently it is not possible to manipulate the pre-processing algorithms with the help of external Graph.xml file.

Sentinel-2 pre-processor is based on standard open source software components: ESA Sen2Cor[11] (recommended version 2.5.5) and GDAL/OGR version 2.x . Sen2Cor older versions (e.g. 2.3.x) are

3 https://www.brockmann-consult.de/beam/doc/help/general/BeamDimapFormat.html
4 https://www.ubuntu.com/download/server
5 http://step.esa.int/main/toolboxes/snap/
6 http://www.gdal.org/gdal_utilities.html
7 http://www.gdal.org/ogr_utilities.html
8 https://github.com/senbox-org/snap-desktop
9 https://github.com/senbox-org/s1tbx/tree/6.x
10 https://github.com/senbox-org/s2tbx/tree/6.x
11 https://step.esa.int/main/third-party-plugins-2/sen2cor/

based on multi-platform Python 2.7 implementation, and Anaconda2[12] had to be installed in the users [home] directory before deployment of the Sen2Cor software component. This was rather inconvenient due to large size and separate maintenance of the Anaconda2 software component. The current Sen2Cor version (2.5.5) does not require a separate install of Anaconda2.

3.3 Hardware requirements

Binary processors were deployed and tested on virtual machines powered by 64-bit Ubuntu 16.04 LTS Server operating system. Deployment of standard OS and geo-processing software components on the VM does not exceed 10 Gb; however, Sentinel data products demand at least 20 Gb of storage space per processing cycle (i.e., deployment of input product, temporary files and ancillary data). It is assumed that input data repository and storage space for output products located outside the virtual machine. The recommended virtual machine storage capacity is 30 Gb for Sentinel-1 products and 50 Gb for Sentinel-2. Virtual machines for the extraction of water/wetness masks should have a storage capacity of 30 Gb, but it is strongly recommended to use bounding boxes to control virtual machine storage space and consumption of RAM.

Java-based ESA SNAP software offers a multi-core GPT processor, which is used for pre-processing of Sentinel-1 products. Certain image post-processing tasks implemented with GDAL software are programmed to run on multiple cores. We recommend using virtual machines with 4 processor cores for pre-processing of Sentinel-1 products. On the other hand, Sentinel-2 atmospheric correction with Sen2Cor is using only 1 processor core; therefore, to save resources, we recommend to use virtual machines with 1 processor core for pre-processing of Sentinel-2 products. The same type of 1-core virtual machines can be used for the extraction of water/wetness masks.

Consumption of RAM is considerably higher for Sentinel-1/2 image pre-processing; at least 8 Gb of RAM should be allocated for image pre-processing virtual machines (at least 10 Gb RAM recommended for Sentinel-1 pre-processor using Java-based SNAP GPT component). Virtual machines used for the extraction of water and wetness masks can run on rather low memory, however, this parameter depends on the size of processing grid tiles. The optimal settings should be defined by experimental testing, but 8 Gb RAM setting seems to be a reasonable starting value.

3.4 Main features

Sentinel-1 GRD IW polSAR images pre-processing is done on a full scale, including calibration, terrain flattening, and ortho-rectification with SRTM 1 arc-sec DEM. However, the algorithm does not apply any despeckling functions in order to preserve the original resolution and full scale of original details. Accidental speckles are effectively filtered out by statistical aggregation at later processing stages.

Sentinel-1 pre-processor creates a standard Float64 DIMAP product which contains calibrated, terrain-flattened and ortho-rectified VV and VH gamma backscatter values derived from the original SAFE data package. It maintains the original geographic projection (EPSG:4326) and therefore it can be used as input for reprojection into various user-specified metric coordinate systems. It is recommended to store the pre-processed Float64 DIMP products if integration of pre-processed Sentinel-1 products into different regional coverages (in different plane projections) is planned within

[12] https://conda.io/docs/user-guide/install/download.html

a short period of time. Long-term storage may be complicated due to the large size of Float64 products.

Sentinel-1 pre-processor performs the transformation of original polSAR back-scatter values from Float64 data type to unsigned 16-bit integer values (multiplying by 1000) in order to reduce the pre-processed file sizes by 5 times and speed-up the further calculations by operating integers instead of floating-point values. Obviously, this transformation is reversible (dividing integer values by 1000).

Sentinel-1 and Sentinel-2 products are transformed into a metric ("plain") projection, defined by the user. We supply low-resolution PNG preview pictures for production-ready S1/2 geo-tiff images.

Sentinel-2 pre-processor creates 2 additional processing levels, which are generated by Sentinel-2 binary processors. We introduced a transitional processing level L2B to indicate atmospherically corrected Sentinel-2 granules (L2A), transformed from SAFE format into standard GeoTIFF projected into a user-specified EPSG projection. Sentinel-2 L2C product consists of reconstructed Sentinel-2 scenes (either scaled to original size, or clipped with a user-specified processing tile). This is essential in agriculture and forestry applications, as there is a large number of parcels/blocks split into parts by granules.

Sentinel-2 is delivered in standard band combinations of 10, 20, and 60-meter resolution, as they are created by the Sen2Cor processor. However, for agriculture applications, we produce our own cloud/shadow mask using an "aggressive" algorithm.

3.5 Processing workflows

The initial stage of the processing workflow includes selection of input datasets and deployment into “input” locations (folders) within a production environment in such a way that the designated number of processors are provided with an equally balanced number of source products (one input folder per processor).

On dedicated production servers there can be several parallel processing tasks running in parallel if physical hardware resources are available. For Sentinel-2 pre-processing, it is strongly recommended to make sure that L1C and L2A products of the same date are loaded into the same input folders. This will allow processors to detect L2A products and skip the “heavy” processing of atmospheric correction.

It is also important to set up a proper network storage location to collect the processing results, which is called the “output” folder. This location can be used simultaneously by many pre-processors, therefore, in the original design, the SAGRIS processors had built-in capability for scanning the status of the output folder on every iteration in order to adjust parallel processing batches and avoid repetition in processing images which are already dumped into the output folder.

The current version of SAGRIS processors is simplified and designed for the use in a cloud-based HPC environment, which assumes that the parallel processing cluster is managed by an external “broker”, such as RabbitMQ messaging service. To avoid overload of I/O capacity of the output folder while many processors are dumping their processing results in parallel, each processor can be temporarily locking the output folder while dumping files, while the other completed processors are put on hold.

Because this rather primitive self-brokerage of access to the output folder may cause considerable delays of “waiting” processors, we recommend using an additional cron-based SAGRIS collector service, which iterates in an infinite loop and collects the pre-processed images directly from local storage containers on virtual machines. By using an external collector, the efficiency of production can be increased dramatically, as well as the internal network access speed, delivery of the processing results, etc.

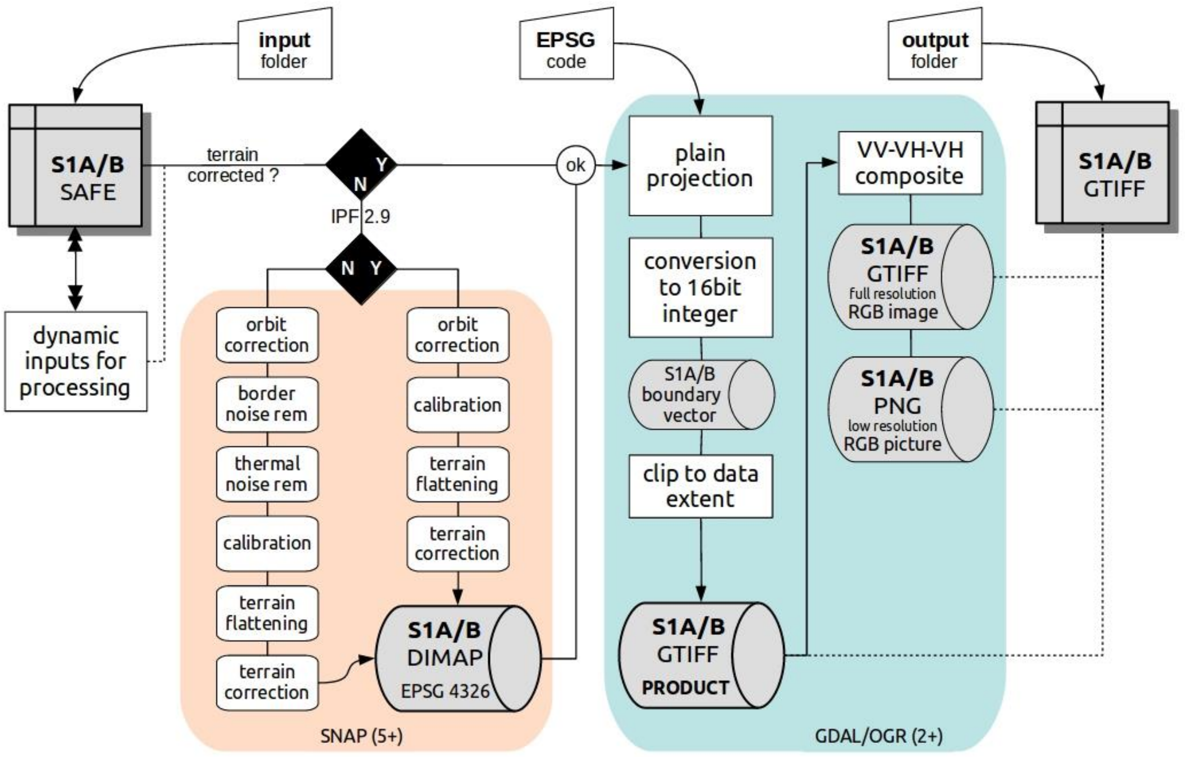


Figure 6 Sentinel-1 pre-processing workflow: load the original SAFE package → calibrate, flatten and orthorectify with SNAP into DIMAP package → transform DIMAP into 16-bit GeoTIFF using GDAL → apply user-specified EPSG projection using GDAL → produce false col

Sentinel-1 processor workflow starts with calibration and ortho-rectification done with SNAP and the outcome is the pre-processed DIMAP data package which contains pre-processed backscatter signal values in Float64 data type and geographic coordinate system. This is a "master copy" of any further derivative products made of this image - a large zip file with "_TC4326" at the end of its name, which means "Terrain-corrected EPSG:4326 projection". The user can keep those large files in the archive or delete them if sufficient processing capacity to re-process the original SAFE packages is available.

The next Sentinel-1 pre-processing step is transformation of the product into Int16 data type (value*1000) and resampling into a user-specified metric projection. This produces a number of files with "_TCxxxx" at the end of the file name (in case of Lithuania that would be “TC3346 - EPSG:3346"). There are 3 such files: a zip file with projected polSAR products and a shapefile delineating the image boundary, then a *_pct full-resolution RGB image useful for visual analysis and WMS services, and a downgraded png picture for the image preview. That is all for S1 products. If projected images are needed for further processing, keep the *_TCxxxx.zip file and delete everything else. This is the only file to be used for the further production.

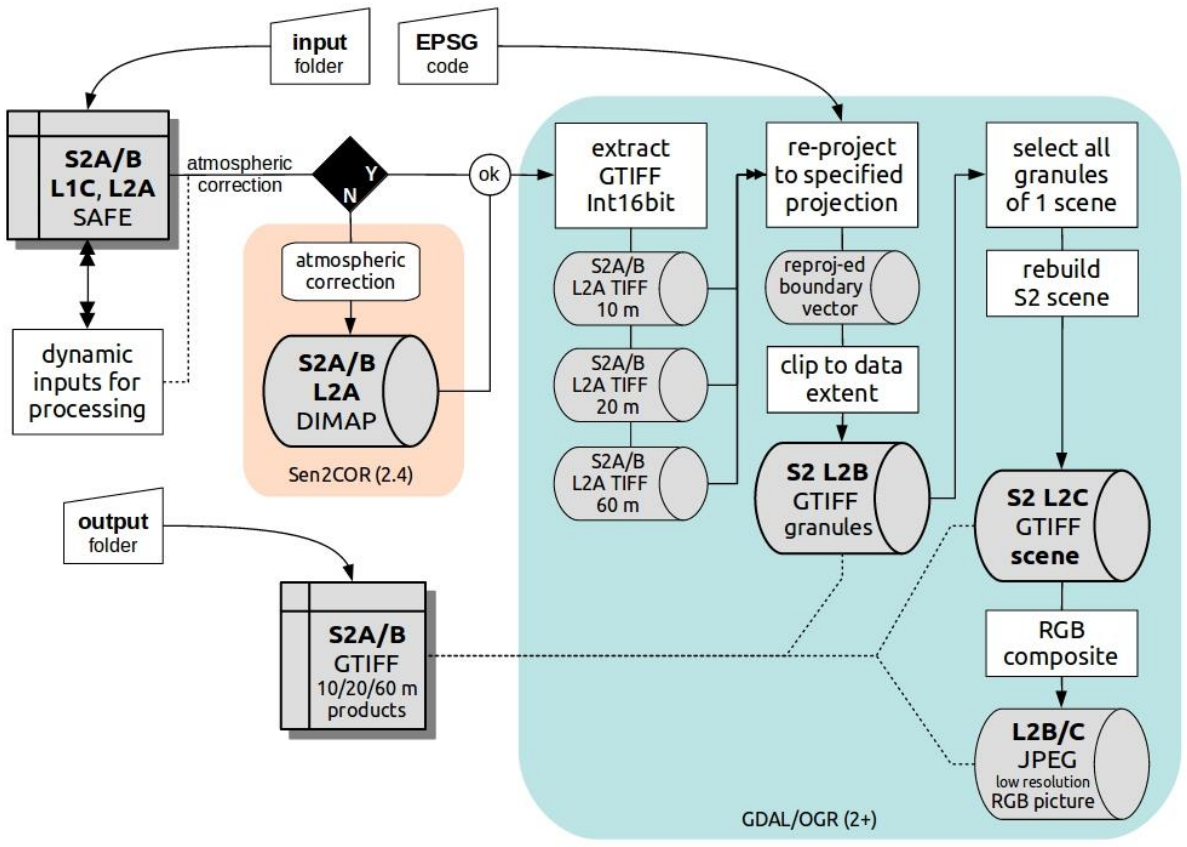


Figure 7 Sentinel-2 pre-processing workflow: load original (orthorectified) SAFE package (a single Sentinel-2 granule) → apply atmospheric correction (if it is L1C product) with Sen2Cor and produce an L2A processing level SAFE package → transform SAFE package into 10m, 20m and 60m resolution 16-bit GeoTIFF images using GDAL → apply user-specified EPSG projection using GDAL → produce false-color composite image from spectral bands.

S2 processor does a much more complicated work, as it must deal with several processing levels. You should start processing from MSIL2A products (if there are such in the downloaded batch), and when their processing is completed, start processing the MSIL1C products. If you dump outputs from all the processors into the same folder, they will automatically detect if the incoming L1C products were already processed into L2A and start the processing only in case if L2A products are not produced yet. In any case, either downloaded or locally created L2A products in standard SAFE format will be transformed into separate 10m, 20m and 60m multi-spectral GTIFF images (still in their original UTM projection), which are named as *_10m.zip, *_20m.zip and *_60m.zip files with MSIL2B processing level code.

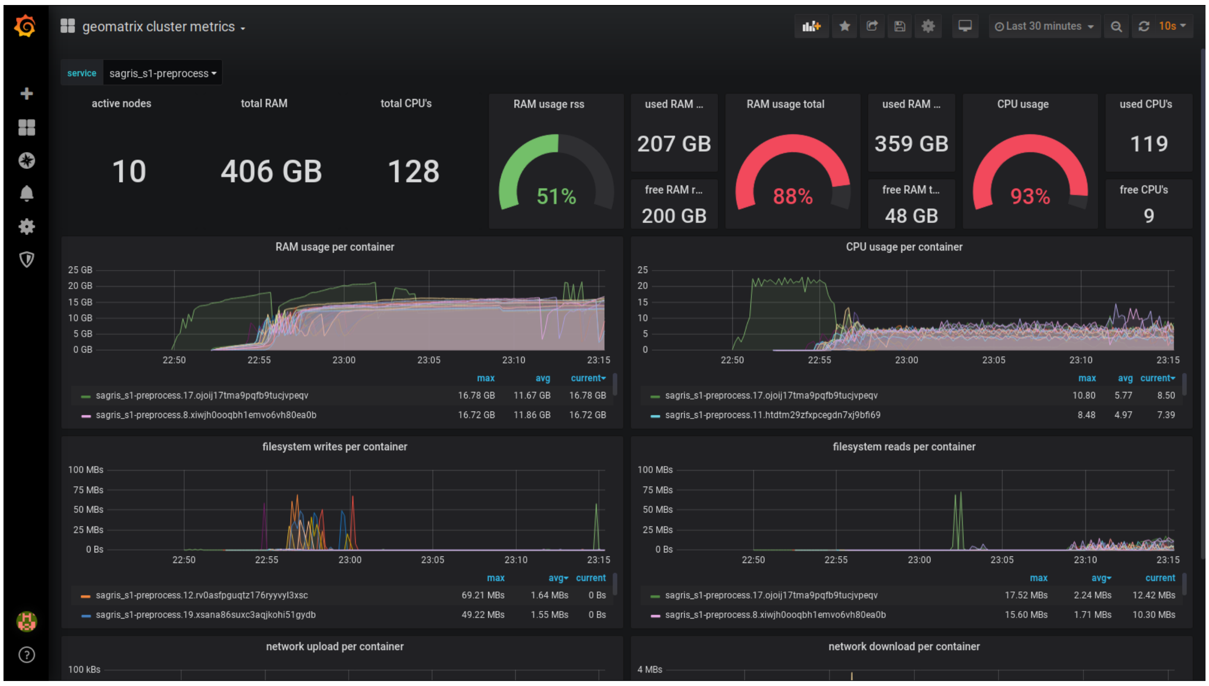
After that, L2B GTIFF images will be resampled into a user-specified EPSG projection, and their file names will get a corresponding EPSG code number after 10m, 20m and 60m notations. The process will also create 2 jpeg preview images: one for MSIL2A product, and the other for the projected MSIL2B product. Spectral bands of MSIL2A and MSIL2B GTIFF images correspond to the standard specification of the Sen2Cor processor: 10m - B2, B3, B4, B8; 20m - B2, B3, B4, B5, B6, B7, B8A, B11, B12. For further extraction of water and wetness masks the workflow will only need projected 10 m and 20 m MSIL2B images (i.e., those zip files with EPSG numeric code at the end of a file name), so it is safe to delete all the other products.

3.5.1 Parallel processing

SAGRIS pre-processors were originally designed to handle one Sentinel-1 or 2 product per processing task (see paragraph 3.7), however scaling up to the parallel processing environment – like a cloud-based DIAS platform – demanded even more "product-centric" software re-design, automated initiation of "batches" containing multiple processing tasks, as well as implementation of a specific parallel processing brokerage solution.

SAGRIS binary processors are developed as Python scripts by GMX team is completely based on cloud-ready docker platform under Linux OS, exploiting standard versions of the popular open source software (GDAL/OGR, ESA SNAP toolbox and GRASS[13] GIS). For automated tasking and management of parallel processing tasks we use open source RabbitMQ[14] software, which is one of the most popular open source message brokers. Remote control and monitoring of the production processes is done with an excellent on-line interface offered by Grafana[15] software, as shown in the following real-life SAGRIS production monitoring example (Fig. 8).

Parallel processing environment is currently deployed on GMX production system with 35 TB dedicated storage space (planned up to 100 TB storage space to be installed before the end of the project), 130 high performance processors (planned up to 250 CPU cores) and 400 Gb total RAM (planned up to 1 Tb). The production system has a 1 Gb optical fibre internet connection, which can download Sentinel-1 products from Copernicus Open Access Hub[16] within 10-15 sec.



RAM usage per container

	max	avg	current
sagris_s1-preprocess.17.ojoij17tma9pqfb9tucjvpeqv	16.78 GB	11.67 GB	16.78 GB
sagris_s1-preprocess.8.xiwjh0ooqbh1emvo6vh80ea0b	16.72 GB	11.86 GB	16.72 GB

CPU usage per container

	max	avg	current
sagris_s1-preprocess.17.ojoij17tma9pqfb9tucjvpeqv	10.80	5.77	8.50
sagris_s1-preprocess.11.htdtm29zfxpcegdn7xj9bfi69	8.48	4.97	7.39

filesystem writes per container

	max	avg	current
sagris_s1-preprocess.12.rv0asfpguqtz176ryyvyl3xsc	69.21 MBs	1.64 MBs	0 Bs
sagris_s1-preprocess.19.xsana86suxc3aqjkohi51gydb	49.22 MBs	1.55 MBs	0 Bs

filesystem reads per container

	max	avg	current
sagris_s1-preprocess.17.ojoij17tma9pqfb9tucjvpeqv	17.52 MBs	2.24 MBs	12.42 MBs
sagris_s1-preprocess.8.xiwjh0ooqbh1emvo6vh80ea0b	15.60 MBs	1.71 MBs	10.30 MBs

Figure 8 Grafana web interface used for monitoring SAGRIS parallel processing tasks, managed with RabbitMQ messaging system.

[13] https://grass.osgeo.org
[14] https://www.rabbitmq.com
[15] https://grafana.com/grafana
[16] https://scihub.copernicus.eu

Tasking of Sentinel products automated downloads from Copernicus API Hub is currently done by automated scripts automatically launched at fixed times (usually in late night hours) by the server for designated areas, provided as WKT polygons. Tasking of products download and processing for the PARSEC clients will require users login and manual selection of products at the Copernicus Open Access Hub, then sending their saved "carts" (products.meta4 files) to GMX service operator by e-mail. Pre-processing of Sentinel-1 products normally takes at least 30 minutes, so there will be no on-the-fly interactive pre-processing option. To simplify area-based tasking, by the end of August 2020 we will develop a web-based web interface for products selection and tasking. This development will stimulate a wider use of Sentinel-1 products by PARSEC community and facilitate SAGRIS migration to DIAS platform.

3.6 Operating instructions

Sentinel-1 and Sentinel-2 binary processors are designed to work as stand-alone binary programs that have no GUI or web interface but can be integrated into automated processing scripts with input parameters set directly or defined as variables. It is recommended to create a separate production folder (e.g., TEMP) in the [home] area and place the binary processors with executable permissions inside it.

Different binary processors can be deployed inside the same production folder and executed inside it as separate processes if hardware capacity is available. Each processing batch (implemented by different processors) will have its own temporary folder inside the production folder. In case the processing batch was completed successfully, those temporary folders will be wiped out. Otherwise – if the processing of a certain batch failed – it is possible to see the last temporary products and do a quick assessment of the potential problems.

Sentinel-1 and Sentinel-2 binary pre-processors (s1_sagris_p1.pyc and s2_sagris_p1.pyc) are launched in the same way with the same set of input parameters:

- An obligatory -i command line parameter followed by a full path to an "input" data location (relative to the local or network file system);
- An obligatory -o command line parameter followed by a full path to an "output" data location (relative to the local or network file system);
- An obligatory -p command line parameter followed by EPSG projection code for all pre-processed Sentinel-1/2 products in the same processing batch;
- An optional -z command line parameter to initiate automated checks of the downloaded ZIP files with the original Sentinel-1/2 SAFE data packages (this option is recommended, as occasional ZIP file errors may cause errors and interrupt automated processing batches).

Typical examples of commands to launch Sentinel-1 and Sentinel-2 pre-processors (if input and output folders are located on a local file system) are as follows:

```
python s1_sagris_p1.pyc -i $HOME/input/S1 -o $HOME/output/S1 -p 3035 -z
python s2_sagris_p1.pyc -i $HOME/input/S2 -o $HOME/output/S2 -p 3035 -z
```

Sentinel-2 image aggregator binary processor (s2_sagris_p2.pyc) are launched with the following set of input parameters:

- An obligatory -i command line parameter followed by a full path to an “input” data location (relative to the local or network file system). In a continuous automated processing scenario, the input location of the second processing phase (extraction of masks) should be the same as the output location of the first stage (image pre-processing);
- An obligatory -o command line parameter followed by a full path to an “output” data location (relative to the local or network file system). Storage space requirements for the output location of the second processing stage are minimal, compared to that of the first stage, and the overall processing time is considerably shorter.

3.7 SAGRIS rasdaman platform

Pre-processing of Sentinel products with SAGRIS tools is a heavy computing task, but it is even more demanding for the storage space, because after pre-processing Sentinel SAFE packages are transformed into production-ready GeoTiff images, thus demanding roughly the same amount of additional storage space as their source products. This problem is particularly notable for Sentinel-1 polarimetry SAR products because of high frequency of their time series.

SAGRIS pre-processing tools are designed for maximum efficiency in terms of processing speed, resources and physical size of the pre-processed images. In particular, we transform Sentinel-1 products from the original 64-bit float data type to 16-bit integer, which reduces the size of pre-processed images by 5 times (at least), leaving the option of automated backwards transformation whenever needed. Nevertheless, deployment of a substantial Sentinel-1 polSAR data cube on DIAS platform would be expensive and highly unsustainable after the end of the project, if there is no viable business model in place with dedicated funding to sustain that collection on-line presence on the DIAS platform.

For the current operational testing phase GMX offered its production system as a back-up storage container for the PARSEC Sentinel-1 Data Cube, which will be published on-line as a rasdaman service, connected to the Big Data Toolbox operated on a DIAS platform. GMX rasdaman server is already operational and can be accessed at http://parsec.landimage.info/rasdaman/ows with around 10 000 pre-processed Sentinel-1 products from a series of countries inside and outside the EU, ready to use for demonstration and testing purposes. This collection will be further expanded following the requests of PARSEC users.

We are building an efficient and easily expandable infrastructure for deployment of PARSEC Data Cube rasdaman platform. A dedicated rasdaman 10 enterprise server currently has 15 Tb physical storage on a hardware RAID, where the most frequently used Sentinel-1 products will be placed. It is powered by 8-core CPU with 32 Gb RAM and Ubuntu 18.04 LTS Linux OS. The main server will be connected to a series of large capacity storage servers (up to 20 Tb each), connected through an isolated physical network and mounted into the file system of the main server through dedicated NFS shares:

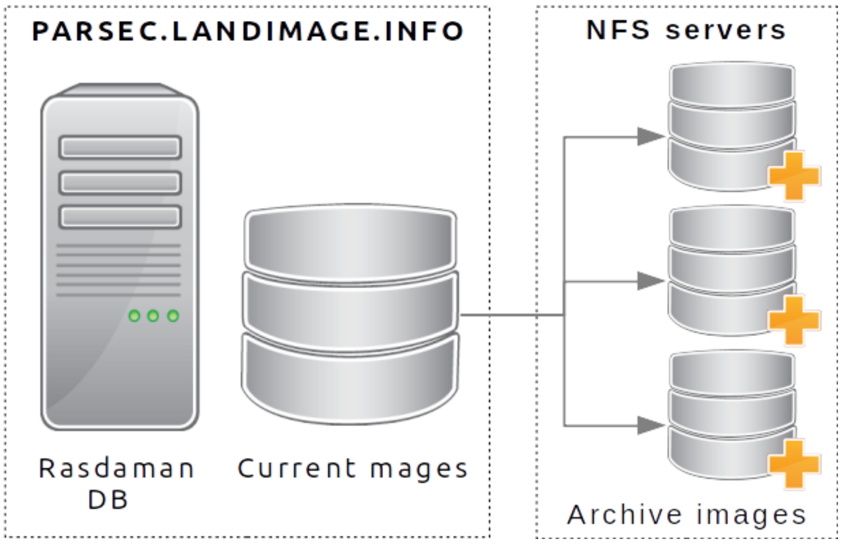


Figure 9 SAGRIS rasdaman platform with main server http://parsec.landimage.info and a series of dedicated NFS servers for deployment of low priority pre-processed images.

Back-up storage servers will host less-demanded (archive) data, which will be gradually removed from the main rasdaman server. This flexible distributed storage infrastructure will gradually expand the Big Data Toolbox service and will help to balance and optimize the overall service performance based on the actual data demand. It will also act as a testing ground for deployment of pre-processed Sentinel-1 datasets, which are currently stored on DIAS platforms, but actually fall out of any practical use due to insufficient processing level.

4 Operational Big Data Toolbox Service

An initial prototype service was temporarily established on GEOMATRIX-owned server with access to Sentinel 1 and 2 data which has been pre-processed by SAGRIS. A demo website to showcase some of the capabilities of the service has been implemented as well by rasdaman (Figure 10). Due to hardware failure the installation setup was lost and this prototype service is not available anymore.

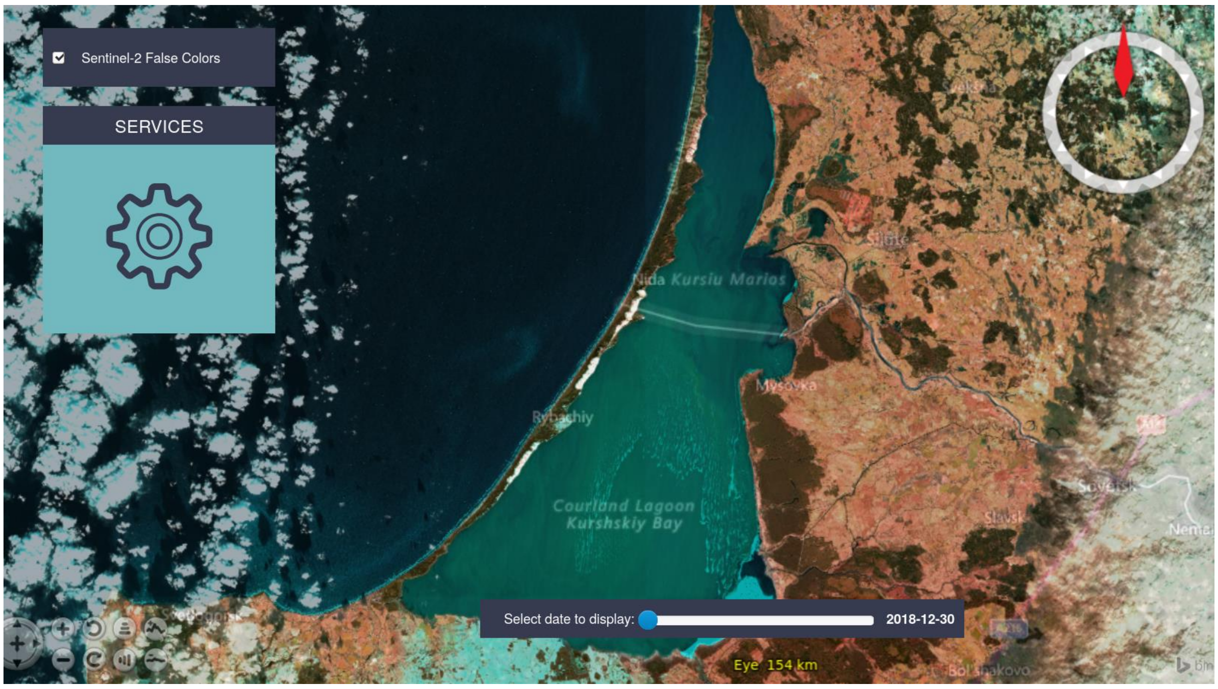
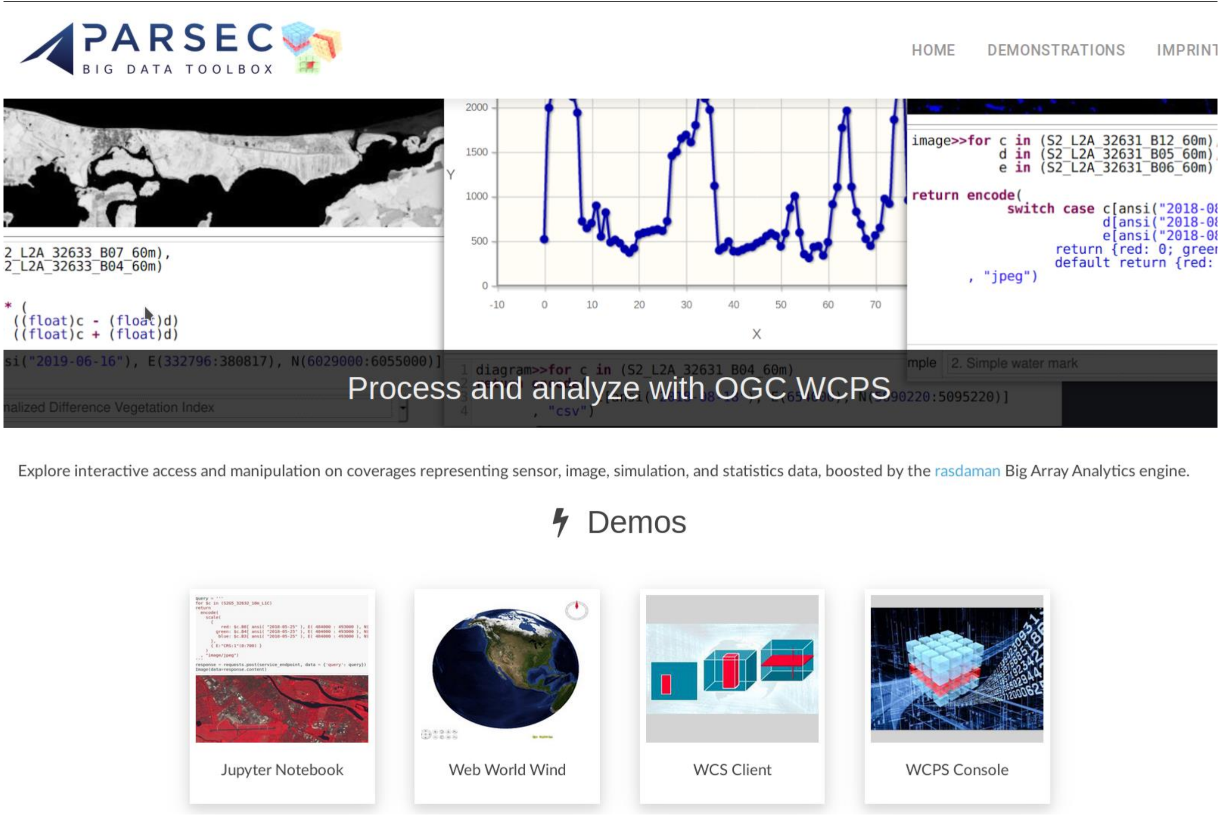


Figure 10 Big Data Toolbox demo entry page showcasing the WMS service on 3D time-series Sentinel-2 data.

Meanwhile, the operational Big Data Toolbox service has been established on the Mundi DIAS platform, available at https://mundi.rasdaman.com/. In the following subsections we describe the service setup, available data and usage.



```
image>>for c in (S2_L2A_32631_B12_60m),
           d in (S2_L2A_32631_B05_60m),
           e in (S2_L2A_32631_B06_60m)

return encode(
        switch case c[ansi("2018-0
                    d[ansi("2018-0
                    e[ansi("2018-0
              return {red: 0; gree
              default return {red:
       , "jpeg")
```

```
2_L2A_32633_B07_60m),
2_L2A_32633_B04_60m)

* (
 ((float)c - (float)d)
 ((float)c + (float)d)
si("2019-06-16"), E(332796:380817), N(6029000:6055000)]
```

```
diagram>>for c in (S2_L2A_32631_B04_60m)
```

Process and analyze with OGC WCPS

Explore interactive access and manipulation on coverages representing sensor, image, simulation, and statistics data, boosted by the rasdaman Big Array Analytics engine.

Figure 11 Operational Big Data Toolbox page with Demos and description of available data.

4.1 Server setup

The main Big Data Toolbox service is deployed on a memory-optimized Elastic Cloud Server (m3.2xlarge.8) in the Open Telekom Cloud via Mundi DIAS since 2020-03-06. The server has the following specification:

- 8 vCPUs @ 3.00GHz
- 64 GB RAM
- 300 GB system disk, Ultra-high I/O
- 1.5 TB data disk, Ultra-high I/O
- 100 Mbit/s network speed
- Elastic IP address 80.158.6.187
- Ubuntu 18.04 OS

The main software driving the Big Data Toolbox is rasdaman enterprise v10.0, which was installed from the standard packages for Ubuntu 18.04.

Additionally, Geomatrix is operating a smaller rasdaman installation dedicated to local pre-processed Sentinel-1 datacubes. This service is federated with the main Big Data Toolbox, so its data is automatically available on the Mundi service as well. The server has the following specification:

- 8-core CPU
- 32 GB RAM
- 15 Tb physical storage on a hardware RAID

- Large capacity network storage of 20 Tb+
- Ubuntu 18.04 OS

4.2 Available data

This Section lists the datacubes offered by the Big Data Toolbox. Mostly the data is confined to what is made available by the DIAS, with exceptions on additional data that was explicitly requested by the beneficiaries (e.g. DEM).

The Big Data Toolbox itself offers more than 15 PB of data spread over ~2300 datacubes. Through the EarthServer Datacube Federation the data offered reaches altogether more than 20 PB.

4.2.1 Sentinel-1

4.2.1.1 Global datacubes with on-demand pre-processing

Sentinel-1 generally requires time-consuming pre-processing in order to build an analysis-ready data cube out of it, as well as expensive disk storage. However, for the Big Data Toolbox rasdaman managed to establish a datacube building procedure that allows to shift the time-consuming pre-processing to an on-the-fly calculation happening during the data retrieval stage when users make queries to the system. This enabled registering and offering Petabytes of Sentinel-1 data through the Big Data Toolbox with only a small penalty of slower data access (less than one minute per one whole scene). As the data is registered and loaded on demand from the DIAS online S3 storage, the datacube could be established with minimal local disk space usage of around 0.5 TB, similar as in the case of Sentinel-2 datacubes.

Datacube details:

- Temporal extent: 2018-01-01 - 2020-06-30
- Spatial extent: global
- Coordinate Reference System: EPSG:4326
- Naming scheme: S1_${product}_${modebeam}_${polarisation}, e.g. S1_GRD_IW_VV
    - ${product} - GRD or SLC
    - ${modebeam} - IW, EW, WV, S1, S2, S3, S4, S5, or S6
    - ${polarisations} - VV, VH, HH, or HV

4.2.1.2 Local pre-processed datacubes

Several spatio-temporal areas have been pre-processed in order to allow real-time datacube access. They are offered on the SAGRIS datacube at http://parsec.landimage.info/rasdaman/ows and are arranged into several coverages for separate countries. This datacube is federated with the main DIAS service, and hence the data is available through both endpoints.
Datacube naming conventions follow the same structure, e.g. S1_GRD_VH_SAGRIS_LIT_3346_10m, where

- S1 indicates satellite sensor
- GRD – product processing level
- VH/VV – product thematic content

- SAGRIS – pre-processing work-flow (also specification)
- LIT – reference country/region
- 3346 – EPSG code of the pre-processed data loaded into rasdaman database
- 10m – spatial resolution of raster products.

Full list of available datacubes:

- *Lithuania* – temporal extent: full Sentinel-1 time series covering 2015-03-02 – 2020-07-08, spatial extent – whole country, time series – all GRD(H) VV and VH images. This dataset is provided for:
    - Monitoring the environment changes and climate impacts during a wide range of conditions, including normal seasons, extreme droughts and floods, normal, cold and mild winters, spring floods, etc.;
    - Analysis of the land cover change and mapping of land use intensity – detection of permanent grassland, development of transitional woodland, etc.
    - Detection of crops and monitoring crop development;
    - Development and testing of forestry applications.

- *Latvia* – temporal extent: 2019-02-25 – 2019-07-05, spatial extent – whole country, time series – all GRD(H) VV and VH images. This dataset is provided for:
    - Development and testing of forestry applications and sustainable energy (bio-fuel);
    - Development and testing of trans-boundary applications and services in the fields of environment and agriculture;
    - Monitoring of physical (moisture/drought) conditions in coastal and inland wetlands of the northern Europe;
    - Monitoring the environment changes in semi-natural landscape used for extensive agriculture (cattle grazing in particular).

- *Denmark* – temporal extent: 2019-05-01 – 2019-08-29, spatial extent – whole country, time series – all GRD(H) VV and VH images. This dataset is provided for:
    - Monitoring the environment changes in northern Europe grasslands and coastal ecosystems affected by tides;
    - Crop detection and monitoring of hazardous weather impacts on crops cultivated in mild winter conditions;
    - Detection of ships in coastal waters.

- *Azerbaijan* – temporal extent: 2019-03-01 – 2019-07-04, spatial extent – whole country, time series – all GRD(H) VV and VH images. This dataset is provided for:
    - Monitoring the environment changes in large coastal and inland wetland ecosystems;
    - Monitoring seasonal dynamics in agriculture production of mountain regions;
    - Detection of coastal and off-shore oil spills;

- *Uzbekistan* – temporal extent: 2018-03-01 – 2018-11-03, spatial extent – whole country, time series – all GRD(H) VV and VH images. This dataset is provided for:
    - Analysis of large-scale desertification processes;
    - Monitoring seasonal dynamics of water resources in Central Asia rivers;
    - Testing crop detection and smart farming algorithms in irrigated farmland systems with two harvests per season.
- *Spain* – spatial extent: Madrid, Lat[40.0823, 40.6848] and Long[-4.3341, -3.2574] (EPSG:4326)

4.2.2 Sentinel-2

Sentinel-2 data is fully registered in the Big Data Toolbox and automatically fetched from the DIAS online S3 storage during query evaluation.

Multiple datacubes built from the original scenes are available, for each level, UTM zone, band and resolution. In addition, *virtual coverages* which unify UTM zones to global coverages in EPSG:4326 for each band are available.

4.2.2.1 Level-1C

- Temporal extent: 2019-03-01 - 2020-04-15
- Spatial extent: global
- Naming scheme: S2_L1C_${utmCode}_${band}_${resolution}, e.g. S2_L1C_32633_B01_10m
    - ${utmCode} - EPSG code for datacube CRS: 32601 - 32660 (N), 32701 - 32760 (S)
    - ${band} - B01, B02, B03, B04, B05, B06, B07, B08, B8A, B09, B10, B11, B12, TCI and PVI
    - ${resolution} - 10m, 20m, 60m or 320m (band PVI)
- Virtual Coverages: S2_L1C_${band}_${resolution}, e.g. S2_L1C_B01_10m (EPSG:4326)

4.2.2.2 Level-2A

- Temporal extent: 2018-03-01 - 2020-03-31
- Spatial extent: Europe (UTM 32628 - 32637)
- Naming scheme: S2_L2A_${utmCode}_${band}_${resolution}, e.g. S2_L2A_32631_AOT_20m
    - ${utmCode} - EPSG code for datacube CRS: 32628 - 32637 (N)
    - ${band} - B01, B02, B03, B04, B05, B06, B07, B08, B8A, B09, B11, B12, TCI, SCL, CLDPRB, SNWPRB, WVP and AOT
    - ${resolution} - 10m, 20m, or 60m
- Virtual Coverages: S2_L2A_${band}_${resolution}, e.g. S2_L2A_AOT_20m

4.2.3 Sentinel-5p

Sentinel-5p data was reprojected to EPSG:4326 CRS, cropped to an area covering Europe, and ingested in rasdaman. The data was limited in order to fit on the local disk storage available.

- Temporal extent: 2019-01-01 - 2020-05-31
- Spatial extent: Europe, Lat(18.7:80.36), Long(-38.71:59.21) (EPSG:4326)
- Naming scheme: S5p_L2_${product}_${variable}
    - ${product} and ${variable} values are explained in the table below

${product}	${variable}	Description
AER_AI (Aerosol Index)	aerosol_index_340_380	Aerosol index from 380 and 340 nm
	aerosol_index_340_380_precision	Precision of aerosol index from 380 and 340 nm
	aerosol_index_354_388	Aerosol index from 388 and 354 nm
	aerosol_index_354_388_precision	Precision of aerosol index from 388

		and 354 nm
AER_LH (Aerosol Layer Height)	aerosol_mid_height	Height at center of aerosol layer relative to geoid
	aerosol_mid_height_precision	Height at center of aerosol layer standard error
	aerosol_mid_pressure	Air pressure at center of aerosol layer
	aerosol_mid_pressure_precision	Air pressure at center of aerosol layer standard error
CH4 (Methane)	methane_mixing_ratio	Column averaged dry air mixing ratio of methane
	methane_mixing_ratio_bias_corrected	Bias corrected column-averaged dry-air mole fraction of methane
	methane_mixing_ratio_precision	Precision of the column averaged dry air mixing ratio of methane
CO (Carbon Monoxide)	carbonmonoxide_total_column	Vertically integrated CO column
	carbonmonoxide_total_column_precision	Standard error of the vertically integrated CO column
HCHO (Formaldehyde)	formaldehyde_tropospheric_vertical_column	vertical column of formaldehyde
	formaldehyde_tropospheric_vertical_column_precision	random error of vertical column density
NO2 (Nitrogen dioxide)	air_mass_factor_total	Total air mass factor
	air_mass_factor_troposphere	Tropospheric air mass factor
	nitrogendioxide_tropospheric_column	Tropospheric vertical column of nitrogen dioxide
	nitrogendioxide_tropospheric_column_precision	Precision of the tropospheric vertical column of nitrogen dioxide
	nitrogendioxide_tropospheric_column_precision_kernel	Precision of the tropospheric vertical column of nitrogen dioxide when applying the averaging kernel
	tm5_tropopause_layer_index	TM5 layer index of the highest layer in the tropopause
O3 (Ozone)	ozone_total_vertical_column	Atmosphere mole content of ozone
	ozone_total_vertical_column_precision	Atmosphere mole content of ozone error
CLOUD (Cloud)	cloud_base_height	cloud base height assumed in ROCINN retrieval
	cloud_base_height_precision	cloud base height precision assumed in ROCINN retrieval
	cloud_base_pressure	cloud base pressure assumed in ROCINN retrieval
	cloud_base_pressure_precision	cloud base pressure precision assumed in ROCINN retrieval
	cloud_fraction	Retrieved effective radiometric cloud fraction using the OCRA/ROCINN CAL model
	cloud_fraction_precision	Error of the retrieved effective radiometric cloud fraction using the OCRA/ROCINN CAL model
	cloud_optical_thickness	Cloud Optical Thickness using the OCRA/ROCINN CAL model

	cloud_optical_thickness_precision	Error of the cloud Optical Thickness using the OCRA/ROCINN CAL model
	cloud_top_height	Retrieved vertical distance of the cloud top above the surface w.r.t. the geoid/MSL using the OCRA/ROCINN CAL model
	cloud_top_height_precision	Height at center of aerosol layer standard error
	cloud_top_pressure	Retrieved atmospheric pressure at the level of cloud top using the OCRA/ROCINN CAL model
	cloud_top_pressure_precision	Error of the retrieved atmospheric pressure at the level of cloud top using the OCRA/ROCINN CAL model

Table 5 Sentinel-5p product sand variables

4.2.4 EU-DEM

This datacube provides the full European Digital Elevation Model.

- Spatial extent: Y (0:5416000) and X(943750:8000000) in EPSG:3035
- Spatial resolution: 25 m
- Coordinate Reference System: EPSG:3035
- Naming scheme: EU_DEM

4.3 Datacube federation

The Big Data Toolbox has been federated with multiple other rasdaman services in order to expand the data offerings beyond what it could host on single machines within PARSEC. It is a full member of the EarthServer Datacube Federation (Figure 12), which consists of altogether eight international members at the time of writing this report:

- Alfred Wegener Institut
- CODE-DE (German Copernicus data hub)
- CreoDIAS
- Feng-Chia University Taiwan
- Helmholtz-Zentrum Geesthacht
- Jacobs University Bremen
- MundiDIAS
- Forschungszentrum Jülich

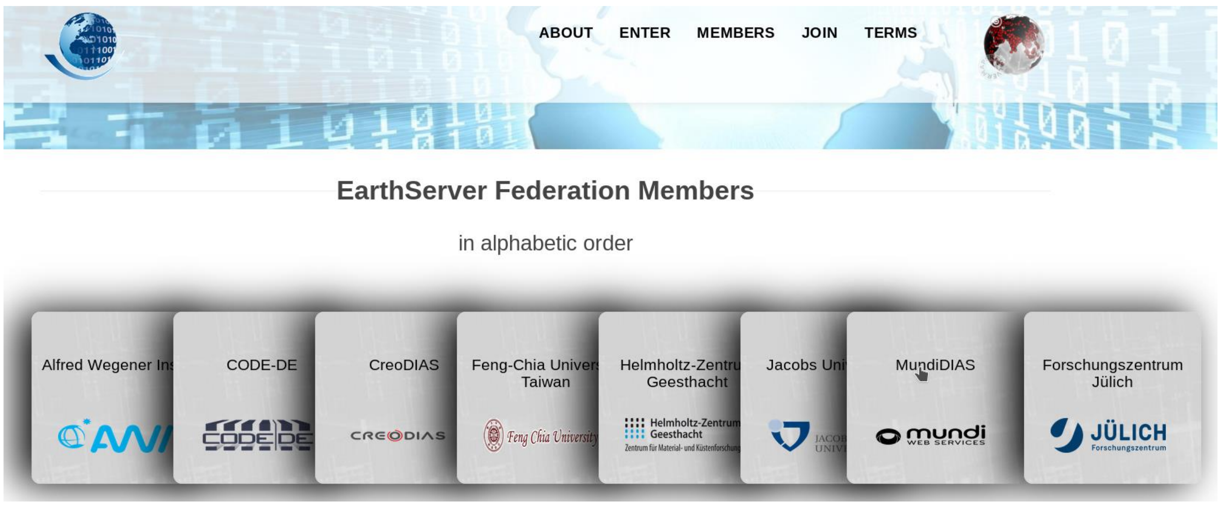


Figure 12 MundiDIAS, aka PARSEC Big Data Toolbox listed in the official EarthServer Datacube Federation members.

Each member brings unique datasets into the federation, and all of them are transparently accessible to users of any of the members. Data from multiple nodes can be freely used and mixed in queries as if it was locally available on a single machine, thereby tremendously simplifying data science. The federation has also been very successful in promoting open data access in the member data centers, although each node has advanced access control capabilities when needed.

4.4 Service development

Building a service to serve the massive Copernicus data available on the DIAS highlighted some areas for improvement, both on conceptual level and in the implementation. The most important features developed for PARSEC are listed below:

- Novel rasdaman query processing engine capable of fully utilizing multicore processors while minimizing memory usage has been finalized in PARSEC, allowing the service to evaluate requests with unprecedented speed.
- Support for quickly registering data hosted on S3 storage, which allowed us to index 6.5 PB of Copernicus data in time to be available to all beneficiaries.
- Support for virtual coverages which unify multiple coverages with different CRS or resolution into a single encompassing coverage; this allowed to present Sentinel 2 data distributed over 120 UTM zones as single global datacubes.
- Support for blacklisting or whitelisting coverages allows to tailor the service UX by hiding data irrelevant to different users.
- Support for fine-grain access control with request-level quotas has been finalized in PARSEC, enabling precise control over service use, which is extremely relevant when a public service holding Petabytes of data is being concurrently accessed by 100s of users.
- Support for multiple rasdaman version in the federation, allowing much more flexible federation networks.

A full list of features implemented, bug fixes and documentation improvements is presented below for completeness:

- add update_db.sh script for migrating users to novel access control mechanism
- fix evaluation of cint16/32 arrays with null values and exporting with gdal

- fix update with floating-point null values and with interval null values
- fix project of subset containing non-materialized tiles
- list roles exempted from trigger should list only roles and not users
- fix type deduction on shift over the projection
- support l and ull type suffixes for 64-bit integer literals
- optimize the use of locally cached tiles in marray evaluation in rasql queries
- fix subset pushing in marray cell expression in rasql queries
- fix compression with GDAL decode
- fix problems with UTM zone handling
- add support for insitu registering overlapping data with null values
- add support for virtual coverages
- add support for multiple versions in rasfed
- add support for coverage blacklisting/whitelisting
- add support for target resolution in project()
- improve federation documentation
- consider rasfed collections in join only in select operations
- support encoding scalar values to csv/json
- fix optimization when iterator reference of a parent marray query expression is used in a child marray expression
- support encoding int64 array with gdal by casting to int32
- fix type deduction for the scale function
- fix mime type format parameters in encode
- fix marray expression rewriting on multiplication in the values clause
- extended RAS_COLLECTIONNAMES to include remote collections available through federation
- fix the performance of overlapping updates
- sentinel 1 SLC custom recipe should not set 0 as default null value in wcst_import
- fix return code of petascope_insertdemo.sh
- fix update of coverages metadata and OWS metadata in wsclient
- petascope supports GDAL 3 to transform coordinates regarding to EPSG CRS YX axes orders
- fix incorrect coordinates for clipping after subsetting in petascope
- petascope set the resolutions for XY axes for gdal
- fix parsing of leading spaces in petascope config
- petascope sets null values to rasql encode
- fix parsing scientific notation as an interval in null values in wcst_imports
- add partial updates section to documentation
- petascope clips by polygon return the same bounding box as trimming by polygon's convex hull
- every subset is checked for belonging into array sdom
- fixed the segmentation fault in update with the wrong domain
- fix segfault during mdd insertion when the mdd type is not a domain type
- fix raswct to catch error messages
- petascope transplates proper XY geo coordinates to grid coordinates
- fix updates of floating-point data with interval null values
- fix wcst_import null error if wms_import is missing in the ingredients file
- petascope catches error with invalid coverage's extent in EPSG:4326
- fix start/stop_rasdaman.sh for multiple rasdaman instances
- fix collection's tiling persistence in petascope database
- added version output to all rasdaman scripts and programs
- wcst_import uses localhost CRS in ingredients file

- fix the problem in SECORE to return GML for version 0
- fix wcs_extract recipe to remove files in case of error
- support updating coverage's metadata with petascope admin user
- fix mock:true in wcs_extract recipe of wcst_import
- fix start_rasdaman.sh on ubuntu 20.04
- wsclient shows number of coverages
- WCPS overlay 2D slices without error
- improve petascope and secore properties
- wcst_import with skip:true should continue if input file fails in general recipe
- support text>> widget to wcs-client
- enhance wcst_import for importing with large number of input files
- implement rasql web console for demonstrating and practicing rasql queries on a web page
- migrate wcst_import to support python 3
- axisnames are properly persisted in RASBASE
- documented RAS_INSTALL_PATH and switch in wcps
- fix stopping of wcst_import daemon when exceptions are thrown in wcst_import
- support unlimited size server message
- WMS style should show unencoded < > characters for query in wsclient
- petascope supports importing gdal /vsi full path
- wcst_import support import of Amazon S3 files
- add support for svg and json output of tiling as option for dbinfo()
- wms returns a transparent image with nodata in json
- rasmgr passes -xp options from rasmgr.conf to rasservers
- improve wcst_import analysis performance on netcdf data
- more user-friendly error in PNG encode with color palettes
- fix the evaluation of multiple marray variables in rasql queries
- add support for foldable code blocks in documentation
- memory leak when java client interrupts result data transfer from rasdaman
- fix retry:false in wcst_import
- wcst_import handles colorPaletteTable for non-general recipes
- support customizing rasdaman service script
- implemented support for positionally-independent subsetting
- wsclient supports https
- implement python3 client API for accessing rasdaman https://pypi.org/project/rasdapy3/
- fix wrong coefficients when encoding in netCDF for irregular axis
- petascope loads gdal-java native library from the default path
- improve logging information shown by wcst_import
- wsclient supports management of WMS downscaled collection levels
- allow subsettingCRS with only one X/Y axis in WCS/WCPS subsets
- fixed problem with a bounding box for WMS style by clipping
- fixed cast of float/double to integer
- add WCS/WMS cheatsheets and an owslib example to the documentation
- wcst_import loads global metadata and color table automatically for general recipe
- wsclient hides DeleteCoverage and InsertCoverage tabs and set output format in GetCoverage tab based on coverage's dimensions

4.5 Service usage

For details on how to make use of the Big Data Toolbox we refer the reader to the accompanying document “D3.11 Big Data Toolbox Training Manual II”, especially Section 4.

PARSEC ACCELERATOR

Our Partners

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824478.