Project Title:	Baltic+ BalticAIMS
WP:	WP2: Service Chain Specification
Document Title:	D2.3: Service Delivery Chain Specification
Version:	1.1 Final version
Author(s) and affiliation(s):	Mikko Kervinen, Sampsa Koponen, Jenni Attila SYKE Norman Fomferra, Alicja Balfanz, Tonio Fincke Gunnar Brandt, Martin Böttcher Carole Lebreton, Carsten Brockmann BC Petra Philipson, Susanne Thulin BG
Version history:	1.0 Version for PDR, 16.9.2021 1.1. Final version, 5.10.2021
Distribution:	ESA, Project team, public

Contents

Glossary

API	Application programming interface
CMEMS	Copernicus Marine Environment Monitoring Service
EO	Earth Observation
HELCOM	Helsinki Commission
HDFS	Hadoop Distributed File System
HR	High Resolution
MSI	MultiSpectral Instrument
NRT	Near Real Time
NTC	Non Time Critical
OLCI	Ocean and Land Color Imager
REST	Representational state transfer
S2	Sentinel-2
S3	Sentinel-3
VM	Virtual Machine
WMS	Web Map Service
WCS	Web Coverage Service
WFS	Web Feature Service
WQ	Water Quality

1 Introduction

1.1 Purpose and scope

This document provides a comprehensive technical description of the end-to-end data acquisition, processing, analysis, delivery and integration. Topics covered in this document are

- Data ingestion and conversion
- Data processing
- Data services and user data access

The other documents generated in WP2 are:

- *D2.1: Service Portfolio Definition:* Contains descriptions of the showcases
- *D2.2 Data and Platform Provisioning Plan:* Contains descriptions of datasets, system structures, interfaces and deployment strategies required to implement the user service portfolio defined in D2.1.

The other documents cover requirements for the data to be served and the complete list of input datasets, and the show cases and their demonstration.

1.2 Document overview

After this formal introduction

section 2	is an overview over showcases, services, and the functional system elements
section 3	describes how the input data is retrieved and converted into XCube zarr format or GeoDB collections
section 4	identifies the processing chains for water quality processing and SST processing on CalFIN and the setup for the processing experiment starting from Sentinel data on CreoDIAS
section 5	shows how the different interfaces of BalticAIMS TARKKA, WCS and WFS, XCube viewer, and also Jupyter notebooks serve data to users

References can be found at the end of the document.

2 System overview

This section is an overview over showcases, services, and the functional system elements.

2.1 Showcases

The 5 show cases are

- A: Provide EO based information to be used in user legacy systems for spatial planning
- B: Monitor the effects of nutrient flow from the drainage basin to the coastal waters
- C: Monitoring the impacts of coastal activities
- D: Combination of Coastal Zone mapping and CMEMS coastal water quality material
- E: Monitoring of temperature anomalies

Each show case is further elaborated in several user stories with concrete applications. Details are provided in D2.1.

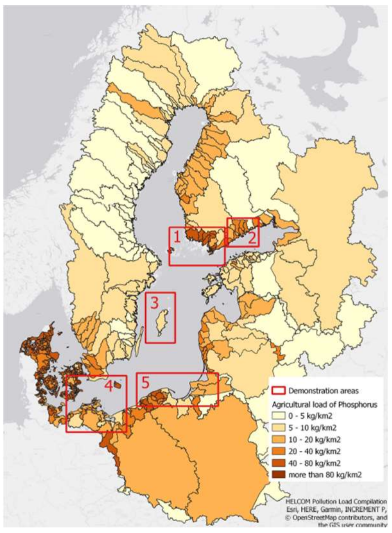


Figure 2-1: BalticAIMS showcase areas

There are 5 areas the show cases will be applied to in different combinations:

- 1. Archipelago Sea, Finland
- 2. Helsinki
- 3. Gotland
- 4. Mecklenburg-Vorpommern
- 5. Poland

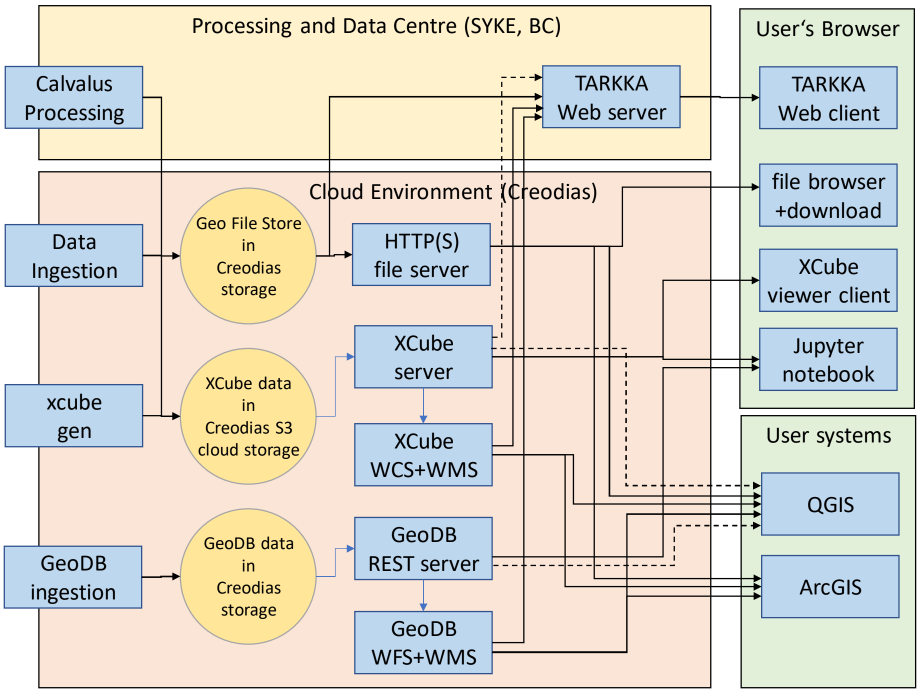
2.2 Services

The BalticAIMS services used by the different show cases are:

- Showcase A makes available EO data for spatial planning using the user systems, mainly GIS. The GIS will read from BalticAIMS data interfaces, in particular WCS and WFS, to access data cubes and GeoDB. Simple static files are served from BalticAIMS geo file server as well.
- Showcase B provides data to analyse the effects of nutrient flow from drainage basins to coastal water. Main service used is TARKKA. The showcase may also use the OGC services provided by BalticAIMS.
- Showcase C provides data to analyse the impact of coastal activities. TARKKA demonstrates the service. GIS interfaces may be used as well.
- Showcase D maps coastal zones. It uses the BalticAIMS data service to include selected layers into the analysis.
- Showcase E makes available EO data to analyse temperature anomalies. It mainly uses TARKKA to demonstrate the service.

2.3 System elements

BalticAIMS data services are based on TARKKA, XCube, and GeoDB. The involved elements are shown in Figure 2-2.



.

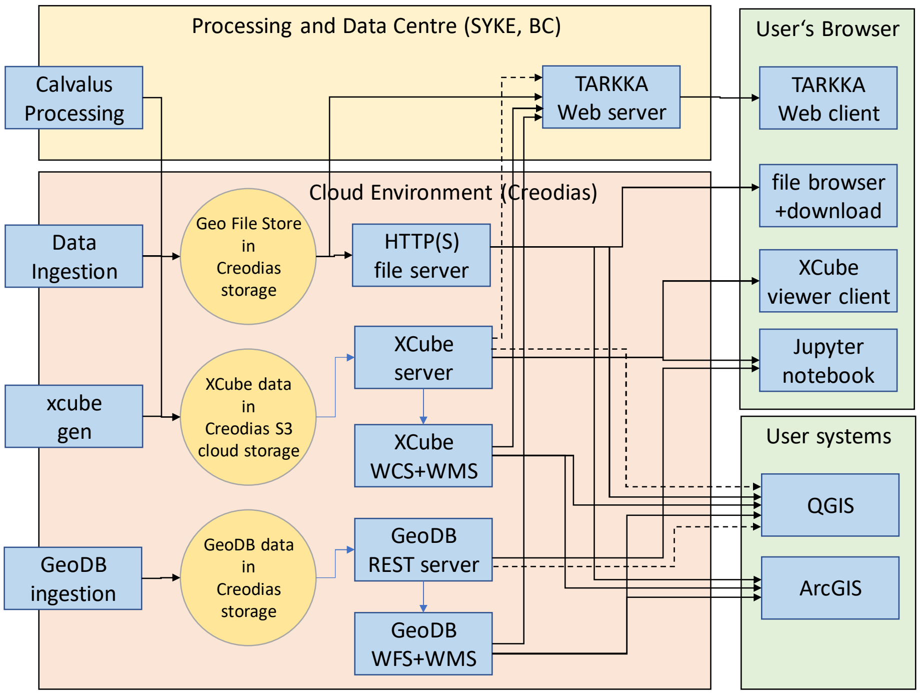


Figure 2-2: BalticAIMS system elements

- Raster time series data is either ingested and prepared with xcube gen, or it is processed by BalticAIMS and stored in XCube zarr format. This data is accessed via XCube server and viewer or via OGC WMS or WCS either by TARKKA or by user GIS applications.
- Feature data is inserted into GeoDB and served via OGC WFS either to TARKKA or to user GIS applications.
- Unstructured or simple file data can also be stored in a structured geo file store and served via HTTP for download or direct access by user GIS applications.

Because WCS is a rather slow and coarse-grained protocol we may switch over to plug-ins for QGIS and TARKKA to access the XCube REST API directly instead (dotted lines). This depends on whether the plug-ins can be developed in time.

3 Data acquisition

This section describes how the input data is retrieved and converted into XCube zarr format or GeoDB collections.

3.1 One-time data retrieval

One-time data retrieval usually is an interactive activity using a browser. As an example, the Corine Land Cover raster dataset is retrieved with the following steps:

- Search for Corine land cover, access address https://land.copernicus.eu
- Log in, create account if necessary
- select Corine Land Cover 100m raster
- Start download, copy the download link
- Use wget to download the raster dataset to a temporary location on the balticcubegen VM on CreoDIAS
- unpack the download item
- Continue with data cube generation

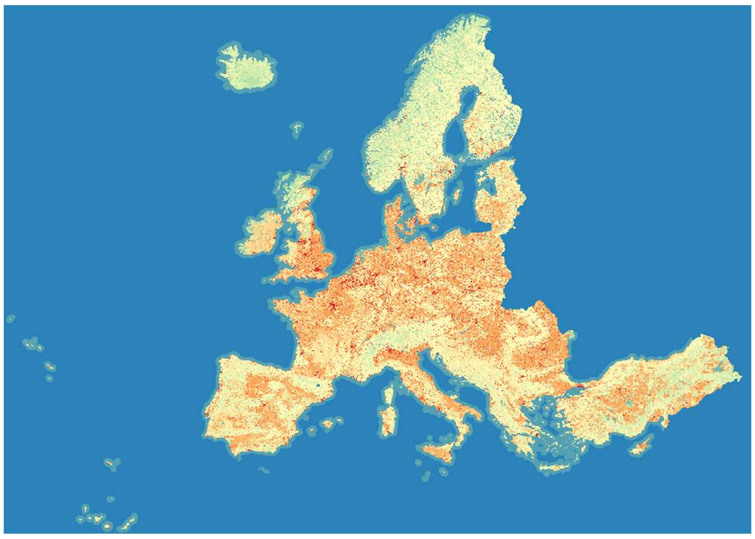


Figure 3-1: Corine Land cover as an example of a downloaded raster dataset

As another example the HELCOM Marine Protected Areas (MPA) vector dataset is retrieved with the following steps:

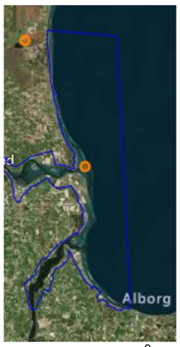


Figure 3-2: Ålborg Bugt, Randers Fjord and Mariager Fjord, shape in dark blue

- The MPA shapefile is downloaded from http://metadata.helcom.fi/geonetwork/srv/eng/catalog.search#/metadata/d27df8c0-de86-4d13-a06d-35a8f50b16fa.
- The shape file are placed at a temporary location on the balticgeodb VM on CreoDIAS.
- Continue with GeoDB collection generation

There may be several datasets that need individual download procedures.

3.2 Systematic data retrieval

Datasets with monthly or daily layers and datasets that are continuously extended can be systematically retrieved. This can either be done with wget -r or with ingestion tools that scan directories for new files and download them. As an example, to download CMEMS SST

- access marine.copernicus.eu
- identify "North Sea/Baltic Sea - Sea Surface Temperature Analysis L3S"
- use a tool (e.g. filezilla) to inspect the directory structure on ftp://nrt.cmems-du.eu/
- use wget -r ftp://nrt.cmems-du.eu/Core/SST_BAL_SST_L3S_NRT_OBSERVATIONS_010_032/DMI-BALTIC-SST-L3S-NRT-OBS_FULL_TIME_SERIE/2021/01 to download the complete month.
- Automate download to download one month after the other.
- download to a temporary location on the balticcubegen VM
- Continue with data cube generation/extension

Project partners may also push systematically generated data directly into specific BalticAIMS object storage locations hosted on Creodias for cube generation.

3.3 Data cube generation

Data cubes are generated with xcube gen from a set of input files using a data cube configuration.

```
output_size: [12594, 14040]
output_region: [9,53,31,66]

output_writer_params:
chunksizes:
lon: 512
lat: 512

output_path: "/mnt/devdata/xcube/hroc-bal-l3d-tur-spm-chl.zarr"

output_variables:
- TUR
- SPM
- CHL
```

Figure 3-3: xcube gen configuration for HR-OC Baltic Sea

```
xcube gen --config hroc-config.yml inputs/hroc/*nc
```

For each cube, the region, resolution/size and variables need to be specified. If the inputs are compliant with NetCDF-CF and have time, lat, lon, then no specific reader needs to be used. Else, the required metadata must be determined by a plug-in to xcube.
After initial cube generation, the cube shall be optimised, pruned, and a pyramid shall be generated with

```
xcube optimise
xcube prune
xcube tiles
```

as described in the xcube documentation. Finally, the xcube server configuration is extended to serve the newly generated cube.

```
...
 - Identifier: S2 HROC BAL
   Title: S2 HR-OC BAL 100m
   FileSystem: local
   Path: "/mnt/devdata/xcube/hroc-bal-l3d-tur-spm-chl.zarr
   Style: S2_msi
...
```

Figure 3-4: xcube server configuration section for the new HR-OC cube

3.4 GeoDB collection generation

GeoDB collections are configured for each vector dataset of BalticAIMS. Ingestion is scripted in Python using geopandas as tool to read the various data formats. The GeoDB API ingests geopandas DataFrame objects into the database.

- Collections are named according to Record Name in the Datasets table.
- Attributes are named by names listed in the Datasets table in column "Variables required", or by their original name if not specified differently.

3.5 Geo file server insertion

The Geo file server is implemented by an Apache web server serving a directory tree of static files. These files are mounted via S3FS as /balticaims in root directory /balticaims/geofiles.

Inserting files into this directory tree means to create a directory and to insert files according to the rule
/balticaims/geofiles/rasterdata/<topic>/<record name>/[<region>/][<temporal coverage>]/<filename>.tif
with

- topic: one of land, water, (extended if helpful)
- record name: entry from Dataset table
- region: optional, if there are separate datasets for different regions, e.g Gotland
- temporal coverage: optional, if there are separate datasets for different periods, e.g. 2018
- filename: either the original file name as provided by the data source, or a suitable name to identify the file

4 Data processing

This section identifies the systematic processing chains for water quality processing and SST processing on CalFIN and the setup for the processing experiment starting from Sentinel data on CreoDIAS.

4.1 Water quality processing on CalFIN

This subsection describes the high-resolution water quality processing chain developed and operated by SYKE

4.1.1 Elements of the Sentinel-2 MSI processing system

- HR-WQ uses the shared SYKE Calvalus cluster that comprises master node, 26 compute nodes, and 1 archive node. The software and data is stored on the Hadoop file system HDFS. A NRT queue is configured with the scheduler to ensure a certain share of the cluster for NRT production in case of high load. The cluster is physically located at the Finnish Collaborative Ground segment at Arctic Space Centre of Finnish Meteorological Institute in Sodankylä.
- Sentinel-2 MSI Level 1C data covering the whole Baltic Sea and Finland is continuously being harvested and archived into HDFS. Usually, the complete coverage is retrieved within 10 hours from the overpass. In case of occasional instabilities and publication delays in Copernicus services, data is harvested as soon it becomes available.
- NRT processing is triggered daily 04:00. Results from the previous day are archived in the cluster file system and transferred to SYKE for manual quality control by SYKE operator.
- Operator checks the preliminary products visually and removes undetected clouds / processing errors in QGIS. When finished, operator triggers final formatting phase where the product is saved as geotiff and published in various locations and services.

4.1.2 Data organisation

The input data (complete S2 tiles) are stored in cluster HDFS file system in daily folders under /calvalus/eodata/S2_L1C/v2/Baltic/{YYYY}/{MM}/{DD}

Processing results are archived in NetCDF-4 format in the HDFS in yearly folders under /calvalus/projects/operative/NRT/HR_WQ/S2/{YYYY}-L3-output/

Last 10 days of processing results are also synchronized to SYKE premises automatically via SCP.

4.1.3 Software structure

The processor software packages are stored in HDFS, too. They are located under /calvalus/software/1.0 by convention. Processor bundle is called EO_HR_WQ_Shadowmasker and it includes individual processors for both S2 MSI and LC8 OLI processing

```
/calvalus/
|--software/
|  |--1.0/
|  |  |--EO_HR_WQ_Shadowmasker-1.3/
|  |  |  |-EO_HR_LC8_WQ-combined-graph.xml
|  |  |  |-EO_HR_LC8_WQ-process.vm
|  |  |  |-EO_HR_S2_60_S2-idepix-graph.xml
|  |  |  |-EO_HR_S2_60_S2-process.vm
|  |  |  |-EO_HR_S2_REFL_WQ-allbands-graph.xml
|  |  |  |-EO_HR_S2_REFL_WQ-process.vm
|  |  |  |-EO_HR_S2_WQ-combined-graph.bak
|  |  |  |-EO_HR_S2_WQ-min-combined-graph.xml
|  |  |  |-EO_HR_S2_WQ-prepare.vm
|  |  |  |-EO_HR_S2_WQ-process.vm
|  |  |  |-EO_HR_S2_bands-bands-graph.xml
|  |  |  |-EO_HR_S2_bands-process.vm
|  |  |  |-EO_HR_WQ_miniconda2.tar.gz
|  |  |  |-auxdata-patch.tar.gz
|  |  |  |-bundle-descriptor.xml
|  |  |  |-common-shadowmaster_calfin.py
|  |  |  |-common-shadowmaster_calfin_template.ini
|  |  |  |-filters.zip
|  |  |  |-mask_auxdata.tar.gz
|  |  |  |-readme.txt
|  |  |  |-snap-6.0.tar.gz
```

- There is one software package for EO_HR_Shadowmasker in version 1.3. This is a shared bundle that includes multiple processors for S2 and LC8 data. It comprises of complete SNAP installation including c2rcc and idepix processors, references to auxiliary data, a miniconda installation, python script for enhanced cloud mask processing, processor wrapper scripts and parameters files. The Sentinel-2 water quality processing chain is implemented in one SNAP GPT processing graph (EO_HR_S2_WQ-min-combined-graph.xml).

4.1.4 Processing system instance structure

The processing system instance controls systematic processing and the progress of processing jobs. It records failure and allows to resume failed steps. The processing system is hosted on a production control node and is structured in a directory tree. In SYKE processing cluster, we use single NRT processing instance with multiple job definitions.

- The runs2waterquality.sh script is scheduled to execute daily, and it contains rules to generate processing requests for new products. Three previous days are processed daily to account for possibly missing tiles/data in the most recent overpasses
- process scheduling is configured via crontab
- There are request templates with default parameters in the etc subdirectory.
- Subdirectories requests and log are filled with new entries at runtime of the service. This allows to trace back all activities of the processing system.

4.1.5 Manual quality control at SYKE

Due to inaccurate automatic cloud masking, SYKE HR water quality products are manually inspected daily during the open water season in Baltic Sea. In the automated production phase, the water quality products (turbidty, cdom, sdt) are masked with certain flag combination based on Idepix cloud detector and in-house enhancements. These preliminary products are then visualized in QGIS where operator can compare the eater quality product with the truecolor visualization of the observation and manually mask out areas where errors remain.

4.1.6 Formatting of the QC data into datacube

Quality-controlled products are transferred back to CalFIN for reformatting and cube generation. Result of the formatting phase is then pushed to BalticAIMS object storage on Creodias for publication.

4.1.7 Systematic processing on Calvalus and SYKE

As explained above, processing is triggered once per day at time when normally the whole input dataset has been harvested into the local processing system archive and so that the processing will be finished before operator starts to work.
- Data harvesting is constantly running background process
- At 4:00 the processing system instance creates processing request that generates daily composite products of the S2 observations from the previous 3 dates.
- The request is submitted to the scheduler of the “cluster”, i.e. to Hadoop YARN. This leads to installation of the software (section 4.3.3) on a compute node, processing with the selected processor and the selected parameters, and archiving of the result water quality product in HDFS (section 4.3.2).
- Once the daily composites are ready, they are automatically archived in the HDFS and staged to a rolling archive folder that is synchronized to SYKE premises every 5 minutes. Process also writes a specific trigger file to the Rolling Archive
- Once the data and the trigger file are transferred to SYKE, a local process runs a simple format conversion from NetCDF-4 to geotiff and stores the files in location where operator can access them via QGIS
- Operator then validates the product and triggers final publishing process manually.

4.2 SST processing on CalFIN

This subsection describes the medium resolution sea surface temperature processing operated by SYKE

Sea surface temperature processing follows similar steps as water quality processing chain.

4.2.1 Elements of the Sentinel-3 SLSTR processing system

- SST processing uses the shared SYKE Calvalus cluster CalFIN (see 4.1.1)

- Sentinel-3 SLSTR Level 1 NRT data covering the whole Baltic Sea is continuously being harvested from Eumetsat CODA service. Usually, the complete coverage is retrieved within few hours from the overpass. In case of occasional instabilities and publication delays in Copernicus services, data is harvested as soon it becomes available in CODA.
- NRT SST processing is triggered daily 04:15. Results are archived in the cluster files system and transferred to SYKE for manual quality control by SYKE operator in the morning after the date of observation.
- Operator checks the preliminary products visually and removes undetected clouds / processing errors in QGIS. When finished, operator triggers final formatting phase where the product is saved as geotiff and published in various locations and services.

4.2.2 Data organisation

The input data (complete S3 SLSTR granules) are stored in cluster HDFS file system in daily folders under /calvalus/eodata/S3_SL_1_RBT/v1/Baltic/{YYYY}/{MM}/{DD}

Processing results are archived in NetCDF-4 format in the HDFS in yearly folders under /calvalus/projects/operative/NRT/SST/S3_SL_1_RBT/v1/SST/{YYYY}-L3-output/

Last 10 days of processing results are also synchronized to SYKE premises automatically via SCP.

4.2.3 Software structure

The processor software packages are stored in HDFS, too. They are located under /calvalus/software/1.0 by convention. Processor bundle is called SLSTR-SST and it the components for SST processing

```
/calvalus/
|--software/
|  |--1.0/
|  |  |-- slstr-sst-2.0/
|  |  |  |-bundle-descriptor.xml
|  |  |  |-org-esa-s3tbx-s3tbx-arc.jar
|  |  |  |-org-esa-s3tbx-s3tbx-idepix.jar
|  |  |  |-slstr-sst-process-graph.xml
|  |  |  |-slstr-sst-rect-process-graph.xml
```

There is one software package for SLSTR-SST in version 2.0. It comprises of required SNAP processor org-esa-s3tbx-s3tbx-arc.jar and processor wrapper scripts. The SST water quality processing chain is implemented in one SNAP GPT processing graph (slstr-sst-process-graph.xml).

For SST the process uses only daily observations that have been acquired after 18:00 UTC i.e., nighttime overpasses.

4.2.4 Processing system instance structure

The processing system instance controls systematic processing and the progress of processing jobs. It records failure and allows to resume failed steps. The processing system is hosted on a production control node and is structured in a directory tree. In SYKE processing cluster, we use single NRT processing instance with multiple job definitions.

- The runsst.sh script is scheduled to execute daily, and it contains rules to generate processing requests for new products. Three previous days are processed daily to account for possibly missing tiles/data in the most recent overpasses
- process scheduling is configured via crontab
- There are request templates with default parameters in the etc/ subdirectory.
- Subdirectories requests and log are filled with new entries at runtime of the service. This allows to trace back all activities of the processing system.

4.2.5 Manual quality control at SYKE

Due to inaccurate cloud masking, SST products are manually inspected daily during the open water season in Baltic Sea. In the automated production phase, the SST products are masked with certain flag combination based on the built-in cloud flags of SLSTR data. These preliminary products are then visualized in QGIS where operator can compare the eater quality product with the truecolor visualization of the observation and manually mask out areas where errors remain.

4.2.6 Formatting of the QC data into datacube

Quality-controlled products are transferred back to CalFIN for reformatting and cube generation. Result of the formatting phase is then pushed to BalticAIMS bucket on Creodias.

4.2.7 Systematic processing on Calvalus at SYKE

As explained above, processing is triggered once per day at time when normally the whole input dataset has been harvested into the local processing system archive and so that the processing will be finished before operator start to work.

- Data harvesting is constantly running process. Both NRT and NTC products are being harvested but the process uses only NRT observations.
- At 4:15 the processing system instance creates processing request that generates daily composite products of the S3 observations from the previous 3 dates.
- The request is submitted to the scheduler of the "cluster", i.e. to Hadoop YARN. This leads to installation of the software (section 4.3.3) on a compute node, processing with the selected processor and the selected parameters, and archiving of the result water quality product in HDFS (section 4.3.2).
- Once the daily composites are ready, they are automatically archived in the HDFS and also staged to a rolling archive folder that is synchronized to SYKE premises every 5 minutes. Process also writes a specific trigger file to the Rolling Archive
- Once the data and the trigger file are transferred to SYKE, a local process runs a simple format conversion from NetCDF-4 to geotiff and stores the files in location where operator can access them via QGIS
- Operator then validates the product and triggers final publishing process manually.

4.3 Water quality processing on BC Calvalus

This subsection describes the medium-resolution processing chain deployed on BC Calvalus.

4.3.1 Elements of the Sentinel-3 OLCI processing system

The Sentinel-3 processing system of Cyanoalert is deployed on BC Calvalus (Figure 4-1). It uses the existing shared processing system and adds just a production control element and the specific data processors.

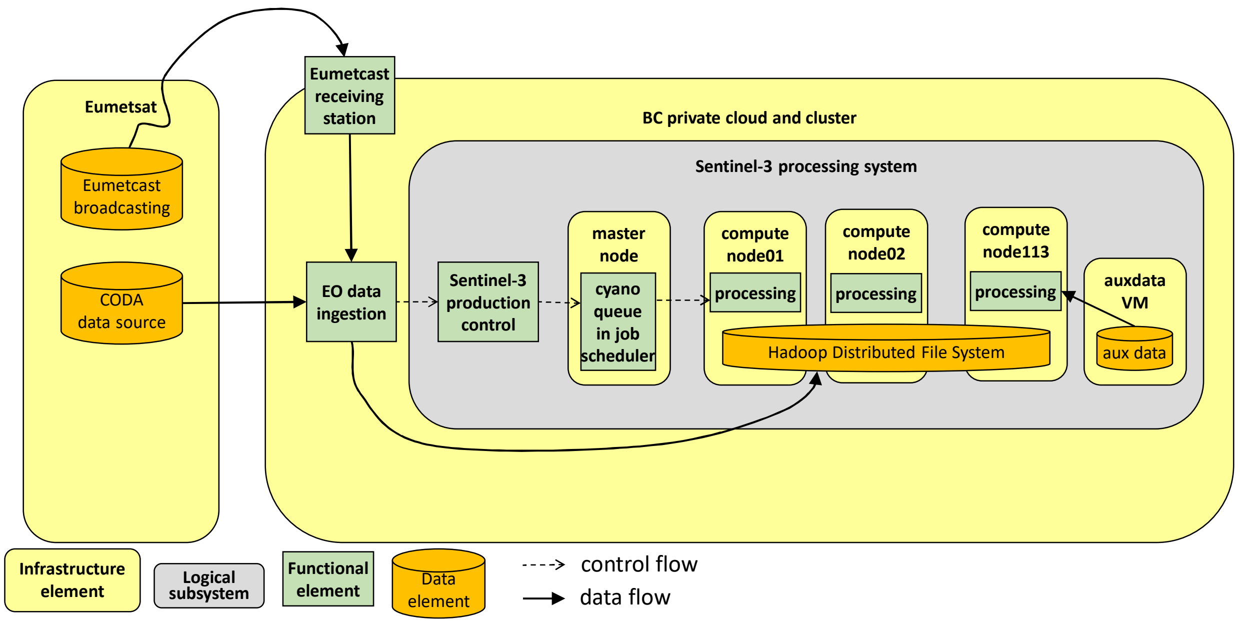


Figure 4-1: Cyanoalert medium resolution processing with Sentinel-3 NRT data from Eumetcast and the shared Calvalus cluster at BC

- Cyanoalert uses the shared BC Calvalus cluster that comprises two redundant master nodes, 113 compute nodes, and 6 archive nodes. The software and data is stored on the Hadoop file system HDFS (instead of S3 on DIAS), and there is a dedicated auxiliary data NFS server here. Else, the software infrastructure is very similar to the one on DIAS, with a YARN scheduler and the Calvalus software. A cyano queue is configured with the scheduler to ensure a certain share of the cluster for Cyanoalert in case of high load.

- Sentinel-3 OLCI Level 1 NRT data is received from Eumetcast with a roof-top antenna at BC. The non-time critical data is downloaded from the CODA server at Eumetsat. In addition, gaps in the NRT data stream are filled with NRT images downloaded from CODA to cope with the inherent instability of the Eumetcast download. About 20 of 450 daily products would else be missing. The Eumetcast reception nevertheless is an advantage as without it the bandwidth of allowed downloads from CODA would limit us to less than half of the 450 products per day.
- The Sentinel-3 production is controlled by a processing system instance that is in fact hosted on the same VM as the one for Cyanoalert Sentinel-2. NRT processing is triggered by the reception of products with the Eumetcast antenna and the subsequent ingestion of the product into Calvalus. As with the Sentinel-2 processing system production control generates jobs on the cluster for new products.

The setup of the Cyanoalert-specific elements of the system is limited to configuration of the processing system instance, deployment of the Sentinel-3 water quality processor package with the processing chain as GPT graph, and the configuration of a cyano queue in Hadoop YARN.

4.3.2 Data organisation

The processing outputs are stored in HDFS in a directory structure below a root /calvalus/projects/cyanoalert as shown in Figure 4-2.

```
/calvalus
|-- projects
|   `-- cyanoalert
|       |-- olci-l2-nrt
|       |   ├── Germany
|       |   │   └── 2020
|       |   │       └── 02
|       |   │           ├── L2_of_S3A_OL_1_EFR____20200201T100234_20200201T100534_20200201T120655_0179_054_236_1980_MAR_O_NR_002.SEN3.nc
|       |   │           ├── L2_of_S3A_OL_1_EFR____20200201T100534_20200201T100834_20200201T120748_0179_054_236_2160_MAR_O_NR_002.SEN3.nc
|       |   │           ├── ...
|       |   │           └── L2_of_S3A_OL_1_EFR____20200206T093539_20200206T093839_20200206T114151_0179_054_307_2160_MAR_O_NR_002.SEN3.nc
|       |   ├── Italy
|       |   │   └── 2020
|       |   │       └── 02
|       |   │           └── ...
|       |   ├── Romania
|       |   │   └── 2020
|       |   │       └── 02
|       |   │           └── ...
|       |   └── Sweden
|       |       └── 2020
|       |           └── 02
|       |               └── ...
```

Figure 4-2: Data organisation of processing results in the Hadoop distributed file system HDFS

- There are different directories for the regions and months. Beyond this level there are the files that have been processed in NetCDF format.
- Systematic processing adds new files into this structure in the current month's subdirectory each time a new input is detected.

4.3.3 Software structure

The processor software packages are stored in HDFS, too. They are located under /calvalus/software/1.0 by convention.

```
/calvalus
|-- software
|   `-- 1.0
|       |-- cyano-1.6
|       |   |-- auxdata-patch.tar.gz
|       |   ├── olci-cyano-l2-2steps-idepix-graph.xml.vm
|       |   ├── olci-cyano-l2-2steps-prepare
|       |   ├── olci-cyano-l2-2steps-process
|       |   ├── olci-cyano-l2-2steps-without-idepix-graph.properties
|       |   ├── olci-cyano-l2-2steps-without-idepix-graph.xml.vm
|       |   ├── c2r_nn_20190225_case12_model_20190202_olci.tar.gz
|       |   ├── idepix-core-7.0.1-SNAPSHOT.jar
|       |   ├── idepix-olci-7.0.1-SNAPSHOT.jar
|       |   ├── libtensorflow-1.9.0-rc1.jar
|       |   ├── libtensorflow_jni-1.9.0-rc1.jar
|       |   ├── org-esa-s3tbx-s3tbx-mphchl.jar
|       |   ├── s3tbx-c2rcc-7.0.1-SNAPSHOT.jar
|       |   ├── s3tbx-fub-wew-7.0.1-SNAPSHOT.jar
|       |   ├── s3tbx-o2a-harmonisation-7.0.1-SNAPSHOT.jar
|       |   ├── s3tbx-olci-radiometry-7.0.1-SNAPSHOT.jar
|       |   ├── s3tbx-rad2refl-7.0.1-SNAPSHOT.jar
|       |   ├── snap-patch.tar.gz
|       |   └── tensorflow-1.9.0-rc1.jar
|       |-- calvalus-2.19.5
|       |   |-- calvalus-commons-2.19-SNAPSHOT.jar
|       |   |-- calvalus-inventory-2.19-SNAPSHOT.jar
|       |   |-- calvalus-processing-2.19-SNAPSHOT.jar
|       |   |-- ...
|       `-- snap-7.0.patch6
|           |-- snap-all.jar
|           `-- ...
```

Figure 4-3: Processor software organisation in HDFS

The comments on software storage on S3 (for Sentinel-2) apply here, too:

- There is one software package for cyano in version 1.6. It comprises .jar files of C2RCC and Idepix for OLCI, references to auxiliary data, and processor wrapper scripts, parameters files, and two graphs (see section 4.3.3). While for Sentinel-2 we managed to provide a single optimised graph for OLCI we have decided to split the even more complex processing graph into one up to Idepix and one for all steps beyond. This does not need that many intermediate nodes in memory at the same time. The memory footprint of the processor gets acceptable with this split.
- There is one software package for Calvalus and one package for SNAP. They belong to the framework of the processing system. They are required to provide separate working directories for concurrent processing tasks, to process inputs with data processors on a Hadoop cluster, and to archive the processing results.
- New versions of processors and additional packages can be installed into this structure without disturbing the operational processing chain.
- The software packages stored on HDFS are automatically deployed (and unpacked) on compute nodes as soon as it is required for processing. This function uses the distributed cache feature of Hadoop to keep the software on nodes up-to-date.

4.3.4 Processing system instance structure

The processing system instance controls systematic processing and the progress of processing jobs. It records failure and allows to resume failed steps. The processing system is hosted on a production control VM and is structured in a directory tree shown in Figure 4-4.

```
cyanoalert-inst
|-- mycyanoalert
|-- olcinrt.py
|-- olcinrt.report
|-- olcinrt.status
|-- etc
|   |-- processing-cht-type.json
|   └-- olci-l2-nrt-template.json
|-- requests
|   `-- ...
|-- log
|   `-- ...
`-- special-requests
    `-- ...
```

Figure 4-4: Cyanoalert processing system instance with scripts and request templates

- The olcinrt script contains the trigger registration to get informed about received and ingested OLCI NRT products, and it contains rules to generate processing requests for new products. .report and .status files inform about the processing progress.
- There are request templates with default parameters in the etc subdirectory.
- Subdirectories requests and log are filled with new entries at runtime of the service. This allows to trace back all activities of the processing system.

This processing system instance is simpler as the Sentinel-2 one as it does not require the access and authentication information for the DIAS, and it is directly triggered by ingestion instead of catalogue queries.

4.3.5 Systematic processing on Calvalus at BC

The dynamic behaviour of the system is depicted in Figure 4-5.

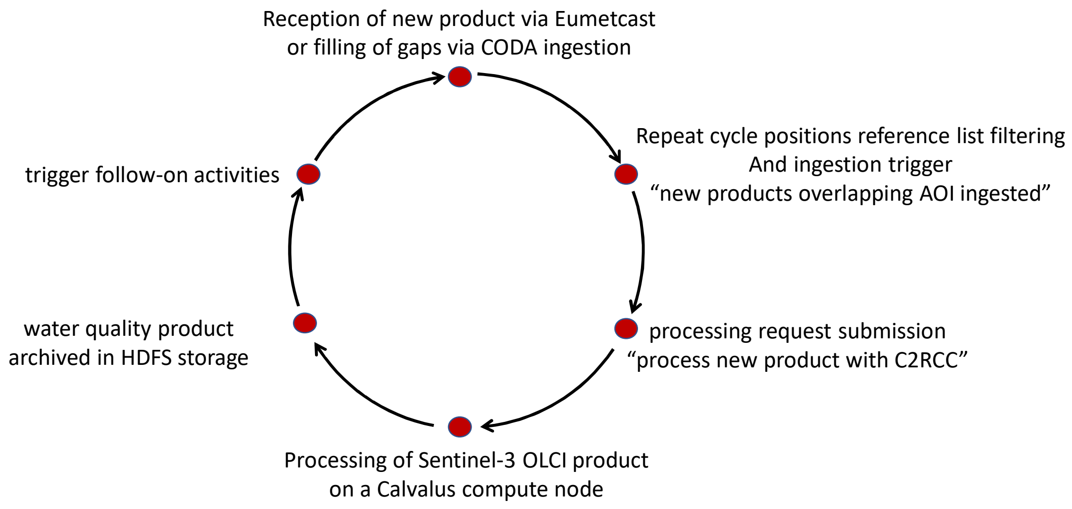

Reception of new product via Eumetcast
or filling of gaps via CODA ingestion
Repeat cycle positions reference list filtering
And ingestion trigger
"new products overlapping AOI ingested"
processing request submission
"process new product with C2RCC"
Processing of Sentinel-3 OLCI product
on a Calvalus compute node
water quality product
archived in HDFS storage
trigger follow-on activities

Figure 4-5: NRT processing event loop with repeated data product reception (antenna)

- Activities are initiated by trigger events from the ingestion system of data from the antenna or from CODA (to fill reception gaps).
- In order to save the time that geo-indexing incoming data usually requires, the geographic filter for the Cyanoalert areas of interest is implemented by translating the intersection problem from a spatial to a temporal query. This makes use of the feature that Sentinel-3 has a repeat cycle after which the geographic situation of what is seen when is the same again. All possible times with intersections within a repeat cycle are pre-computed for the AOI. Each new product is simply compared with the relative times of overlapping scenes within the repeat cycle. This is very efficient.
- For each product a rule generates a processing request using a request template with parameters (section 4.3.4). The request is submitted to the scheduler of the "cluster", i.e. to Hadoop YARN. This leads to installation of the software (section 4.3.3) on a compute node, processing with the selected processor C2RCC and the selected parameters, and archiving of the result water quality product in HDFS (section 4.3.2).
- Indirectly, the extension of the data cube is triggered by the collected new products of a day as follow-on activity.

4.4 Processing environment on CreoDIAS

BalticAIMS will use a Calvalus processing system to process Sentinel data on a DIAS system. This exercise is performed to demonstrate that BalticAIMS is able to process in the cloud, e.g. if this is the place where the larger input datasets are available without the need for download.

A cluster setup tool creates the virtual machines and installs the processing system software on them before jobs are processed on this cluster. This usually is a one-time activity. Figure 4-6 shows the main elements involved in the setup.

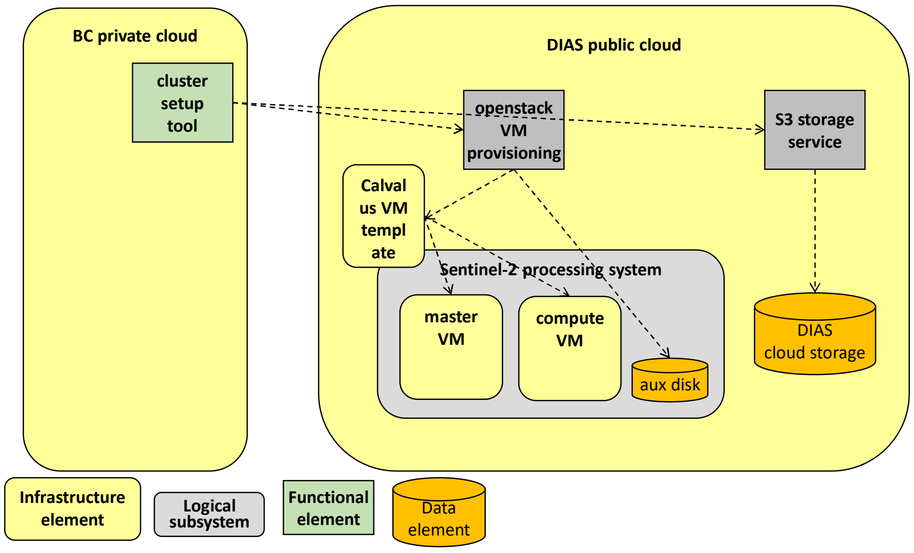
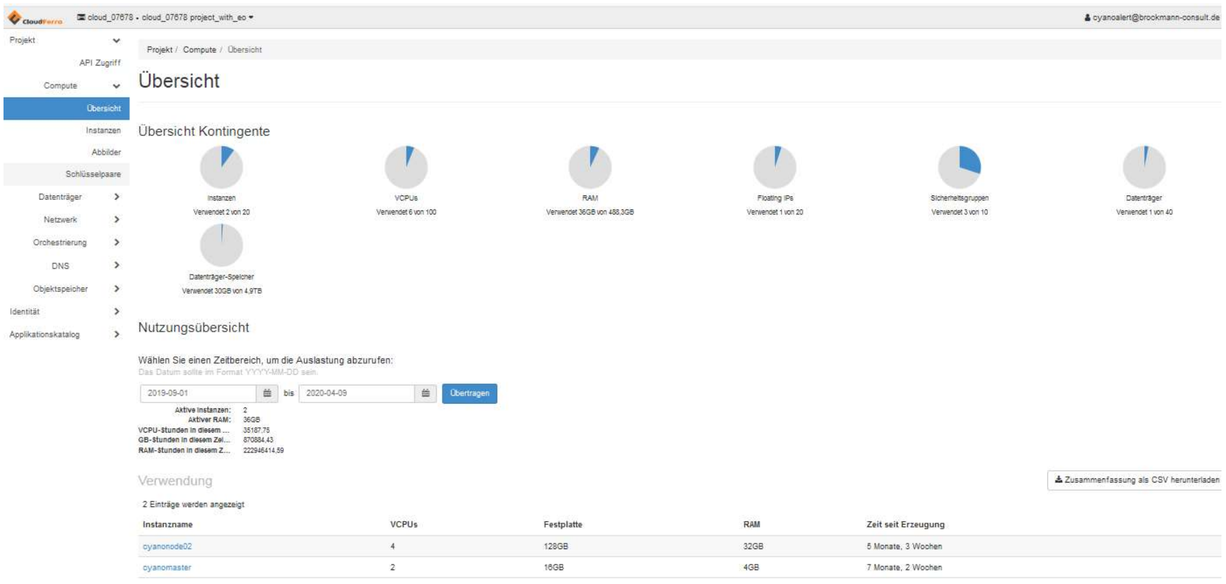


Figure 4-6: Setting up the cluster using the Openstack and S3 cloud storage services of a DIAS

- The cluster setup tool uses a configuration and a software repository.
- It creates a Calvalus VM template that can be used either as master VM or as compute VM.
- It also creates these VMs using the template, further an auxiliary data disk that is mounted to the master VM that also serves as auxiliary data NFS server for the compute nodes.
- It finally creates the cloud storage bucket 'balticaims' (unless done already) that stores processor software packages as well as processing outputs.
- The cluster setup tool uses the client tools of Openstack and S3 to interact with the corresponding DIAS services for VM provisioning and storage service.

The setup is done initially before processing is started. It can also be used to add a compute VM for higher demands, or to remove one if no longer needed. And it could be used to migrate the complete processing system to another DIAS, in particular to Sobloo (Orange) or Mundi (OTC).



Übersicht

Übersicht Kontingente

Instanzen	VCPUs	RAM	Floating IPs	Sicherheitsgruppen	Datenträger	Datenträger-Speicher
Verwendet 2 von 20	Verwendet 6 von 100	Verwendet 36GB von 488,3GB	Verwendet 1 von 20	Verwendet 3 von 10	Verwendet 1 von 40	Verwendet 30GB von 4,9TB

Nutzungsübersicht

Wählen Sie einen Zeitbereich, um die Auslastung abzurufen:
Das Datum sollte im Format YYYY-MM-DD sein.
2019-09-01 bis 2020-04-09

Aktive Instanzen:	2
Aktiver RAM:	36GB
VCPU-Stunden in diesem ...	35187,75
GB-Stunden in diesem Zei...	870884,43
RAM-Stunden in diesem Z...	222946414,59

Verwendung

2 Einträge werden angezeigt

Instanzname	VCPUs	Festplatte	RAM	Zeit seit Erzeugung
cyanonode02	4	128GB	32GB	5 Monate, 3 Wochen
cyanomaster	2	16GB	4GB	7 Monate, 2 Wochen

Figure 4-7: CreoDias cloud dashboard with 2 VMs, a data disk, and the network configuration

The cloud dashboard in Figure 4-7 shows the infrastructure elements on CreoDias, two virtual machines for the master and a compute node, and counters for one external IP, network security rules, and one (auxliliary) data disk.

4.4.1 Data organisation

The processing outputs are stored in DIAS cloud storage in the 'balticaims' bucket. In the bucket the data is below a root /calvalus/projects/balticaims as shown in Figure 4-8.

```
/calvalus
|-- projects
|   `-- balticaims
|       |-- s2-c2rcc
|       |   |-- Archipelago
|       |   |   `-- ...
|       |   |-- Poland
|       |   |   |-- 2021-03
|       |   |   |-- ...
|       |   |   |-- 2021-06
|       |   |   |   |-- S2A_MSIL1C_20210610T100031_N0207_R122_T32TQM_20210610T111156_c2rcc.nc
|       |   |   |   |-- S2A_MSIL1C_20210610T100031_N0207_R122_T33TTG_20210610T111156_c2rcc.nc
|       |   |   |   |-- ...
|       |   |   |   |-- S2B_MSIL1C_20210622T095039_N0207_R079_T33TVF_20210622T115114_c2rcc.nc
|       |   |   |   `-- S2B_MSIL1C_20210622T095039_N0207_R079_T33TXE_20210622T115114_c2rcc.nc
|       |   |   |-- 2021-07
|       |   |   |-- ...
|       |   |   `-- ...
|       |   |-- ...
|       |   |   `-- ...
|       |   `-- Gotland
```

Figure 4-8: Data organisation of processing results in the balticaims cloud storage bucket

- There are different directories for the used algorithms, the region, the month. Beyond this level there are the files that have been processed in NetCDF format.
- Systematic processing adds new files into this structure in the current month's subdirectory each time a new input is detected.

4.4.2 Software structure

The processor software packages are stored in S3 in the balticaims bucket, too. They are located under /calvalus/software/1.0 by convention.

```
/calvalus
|-- software
|   `-- 1.0
|       |-- c2rcc-1.5
|       |   |-- auxdata-patch.tar.gz
|       |   |-- c2rcc-idepix-msi-graph.xml
|       |   |-- idepix-core-7.0.0.jar
|       |   |-- idepix-s2msi-7.0.1-SNAPSHOT.jar
|       |   |-- libgfortran.so.3
|       |   `-- s3tbx-c2rcc-8.0.0-SNAPSHOT.jar
|       |-- calvalus-2.19-SNAPSHOT
|       |   |-- calvalus-commons-2.19-SNAPSHOT.jar
|       |   |-- calvalus-inventory-2.19-SNAPSHOT.jar
|       |   |-- calvalus-processing-2.19-SNAPSHOT.jar
|       |   |-- geo-inventory-0.6.1.jar
|       |   |-- google-s2-1.0.2-20160721.154530-1.jar
|       |   |-- jackson-annotations-2.2.3.jar
|       |   |-- jackson-core-2.2.3.jar
|       |   |-- jackson-databind-2.2.3.jar
|       |   |-- jackson-dataformat-yaml-2.2.3.jar
|       |   |-- jcommon-1.0.16.jar
|       |   `-- jfreechart-1.0.13.jar
|       `-- snap-8.0.0-SNAPSHOT
|           |-- libjhdf5.so
|           |-- libjhdf.so
|           |-- libopenjp2.so
|           |-- snap-all.jar
|           `-- VERSION.txt
```

Figure 4-9: Processor software organisation in the balticaims bucket

- There is one software package for c2rcc shown as an example. It comprises .jar files of C2RCC and Idepix, references to auxiliary data, and a processing graph. The BalticAIMS processor run on CreoDIAS may be a different one.
- There is one software package for Calvalus and one package for SNAP. They belong to the framework of the processing system. They are required to provide separate working directories for concurrent processing tasks, to process inputs with data processors on a Hadoop cluster, and to archive the processing results.
- New versions of processors and additional packages can be installed into this structure without disturbing the operational processing chain.
- The software packages stored on S3 are automatically deployed (and unpacked) on compute nodes as soon as it is required for processing. This function uses the distributed cache feature of Hadoop to keep the software on nodes up-to-date.

4.4.3 Processing system instance structure

The processing system instance controls systematic processing and the progress of processing jobs. It records failure and allows to resume failed steps. The processing system is hosted on a production control VM and is structured in a directory tree shown in Figure 4-10.

```
balticaims-inst
|-- mybalticaims
|-- s2nrt.py
|-- s2nrt.report
|-- s2nrt.status
|-- etc
|   |-- l2-form-cht-type.json
|   |-- processing-cht-type.json
|   |-- s2-c2rcc-template.json
|   `-- s2-extreme-template.json
|-- creo
|   |-- calvalus-creo.config
|   |-- creobaltic.sh
|   |-- creo_key.priv
|   |-- cloud_07678_project_with_eo-openrc.sh
|   |-- dot-s3cfg-creo
|   |-- dot-s3cfg-creodata.template
|   |-- dot-s3cfg-creo.template
|-- requests
|   `-- ...
|-- log
|   `-- ...
`-- special-requests
    `-- ...
```

Figure 4-10: Processing system instance structure with scripts, templates, and DIAS credentials

- The s2nrt script contains catalogue query parameters to inquire new Sentinel-2 L1C products over the areas of interest, and it contains rules to generate processing requests for new products found. .report and .status files inform about the processing progress.
- There are request templates with default parameters in the etc subdirectory.
- There are DIAS credentials and the cluster setup script in the creo subdirectory.
- Subdirectories requests and log are filled with new entries at runtime of the service. This allows to trace back all activities of the processing system.

4.4.4 Systematic processing on DIAS

The dynamic behaviour of the system is depicted in Figure 4-11.

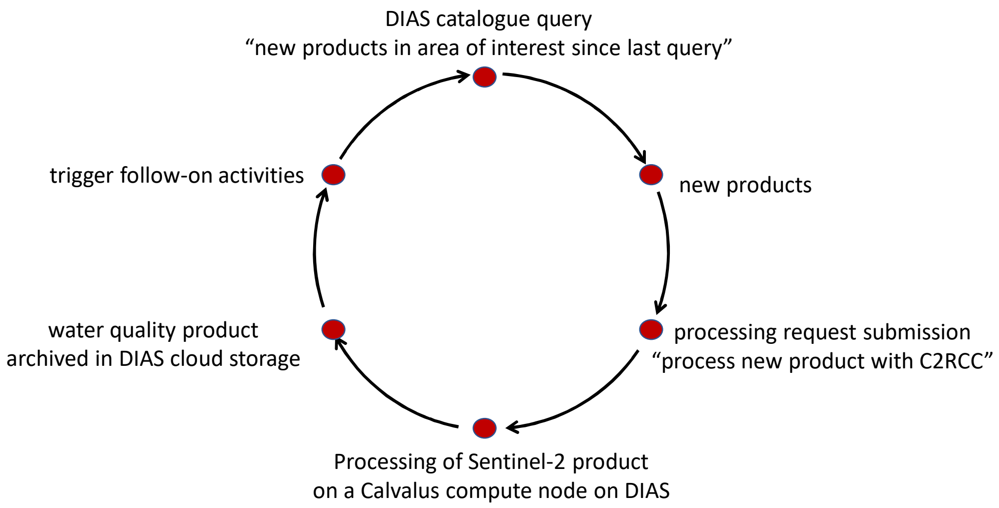

DIAS catalogue query
"new products in area of interest since last query"
new products
processing request submission
"process new product with C2RCC"
Processing of Sentinel-2 product
on a Calvalus compute node on DIAS
water quality product
archived in DIAS cloud storage
trigger follow-on activities

Figure 4-11: NRT processing event loop with repeated catalogue queries and processing requests

- Activities are initiated by a query to the DIAS catalogue. The query asks for products intersecting the AOIs being ingested since the last product that has been processed. The result is a list of products, maybe empty.
- For each product a rule generates a processing request using a request template with parameters (section 4.4.3). The request is submitted to the scheduler of the "cluster", i.e. to Hadoop YARN. This leads to installation of the software (section 4.4.2) on a compute node, processing with the selected processor C2RCC and the selected parameters, and archiving of the result water quality product in cloud storage (section 4.4.1).
- Indirectly, the extension of the data cube can be triggered by the collected new products of a day as follow-on activity.

5 User interaction and data provisioning

This section shows how the different interfaces of BalticAIMS TARKKA, WCS and WFS, XCube viewer, and also Jupyter notebooks serve data to users.

5.1 Use of TARKKA for showcases B, C, D and E

TARKKA is a public web service of the Finnish Environment Institute, where you can browse and view SYKE's open satellite data. The map interface of the service is designed to present both high-resolution (10 m - 60 m) and medium-resolution (300 m - 1 km) satellite data and additional vector datasets.

Tarkka uses Standard OGC interfaces for data access

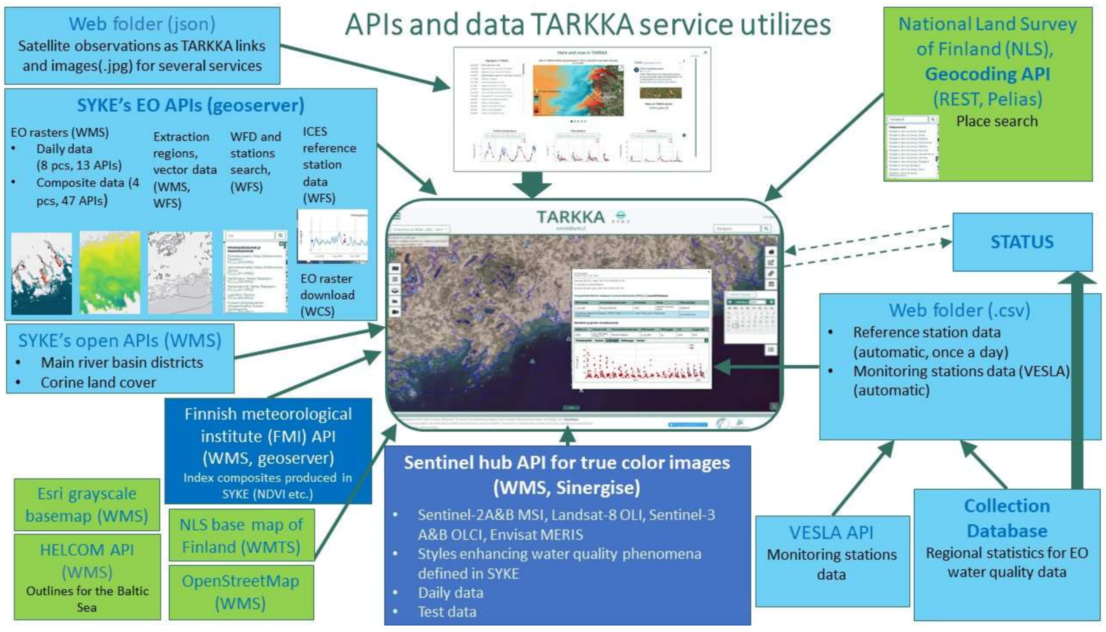
- Web Map Service (WMS) for viewing raster and vector data
- Web Feature Service (WFS) for queries and time series data
- Web Coverage Service (WCS) for downloading data
- Web Map Tile Service (WMTS) for base map

Also, several other interfaces are supported

- REST for geocoding service
- Web folder and csv files for reference station time series
- Web folder and json files for front page links and images

To retrieve information about features and coverages displayed in a map (for example pixel value, region information, in-situ observation metadata) TARKKA uses WMS GetFeatureInfo queries. Most of these interfaces will be provided by the BalticAIMS XCube and GeoDB services.

Dedicated instance of TARKKA will be deployed for BalticAIMS project and hosted on SYKE infrastructure. This instance will connect to the BalticAIMS services on Creodias and provide the relevant data layers and functionality with respect to different BalticAIMS showcases.


APIs and data TARKKA service utilizes
Web folder (json)
Satellite observations as TARKKA links and images(.jpg) for several services
SYKE's EO APIs (geoserver)
EO rasters (WMS)
• Daily data (8 pcs, 13 APIs)
• Composite data (4 pcs, 47 APIs)
Extraction regions, vector data (WMS, WFS)
WFD and stations search, (WFS)
ICES reference station data (WFS)
EO raster download (WCS)
SYKE's open APIs (WMS)
• Main river basin districts
• Corine land cover
Finnish meteorological institute (FMI) API (WMS, geoserver)
Index composites produced in SYKE (NDVI etc.)
Esri grayscale basemap (WMS)
HELCOM API (WMS)
Outlines for the Baltic Sea
NLS base map of Finland (WMTS)
OpenStreetMap (WMS)
Sentinel hub API for true color images (WMS, Sinergise)
• Sentinel-2A&B MSI, Landsat-8 OLI, Sentinel-3 A&B OLCI, Envisat MERIS
• Styles enhancing water quality phenomena defined in SYKE
• Daily data
• Test data
TARKKA
National Land Survey of Finland (NLS), Geocoding API (REST, Pelias)
Place search
STATUS
Web folder (.csv)
• Reference station data (automatic, once a day)
• Monitoring stations data (VESLA) (automatic)
VESLA API
Monitoring stations data
Collection Database
Regional statistics for EO water quality data
Figure 5-1: TARKKA data sources and interfaces

5.2 Use of WCS, WMS, and WFS for showcases A, D, also B, C

The standard OGC interfaces WCS, WMS, and WFS are machine-to-machine interfaces used by GIS applications. As described in chapter 4 of [BalticAIMS D2.2] WCS and WFS allow to add raster and vector layers. The technical interaction between GIS client and WCS or WFS server is shown in Figure 5-2.

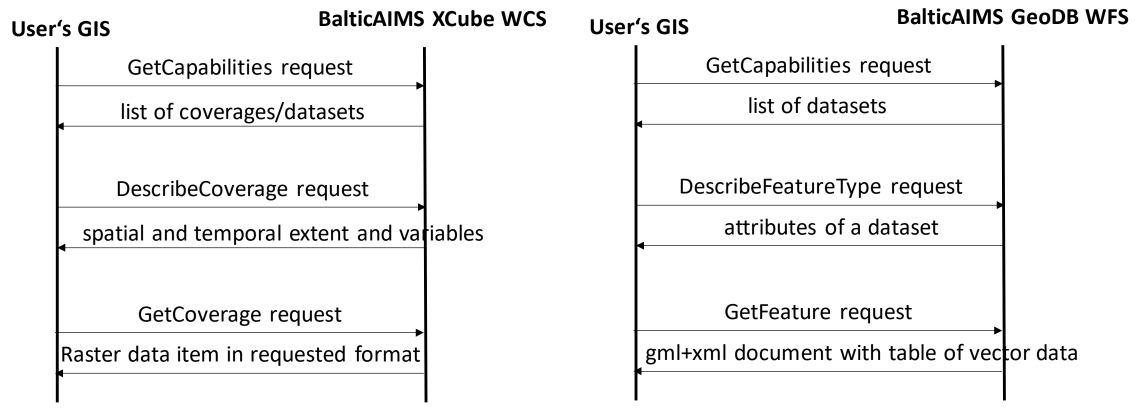


Figure 5-2: Information flow between user's GIS system and BalticAIMS WCS and WFS to select and add BalticAIMS layers to the user's application

- The GetCoverage request is parameterised with the coverage ID (selected from the GetCapabilities response), the fields and the spatio-temporal subsetting (selected and derived from the DescribeCoverage response).
- The GetFeature request is parameterised with the FeatureType name (selected from the GetCapabilities response) and a subset of attributes (selected from the DescribeFeatureType response), a bounding box and an optional filter expression using the attributes.

The results of both types of requests can be directly added as layers in GIS.

- The GetMap request is parameterised with the layer, (selected from GetCapabilties response) and the spatial parameters for projection, resolution, area.

The result can be visualised in TARKKA.

5.3 Use of XCube Viewer for data browsing

The XCube viewer shows the content of data cubes and allows to browse through it together with an initial visualisation and inspection. It lists the datasets available, and the variables contained in datasets. It also shows spatial coverage and temporal extent.

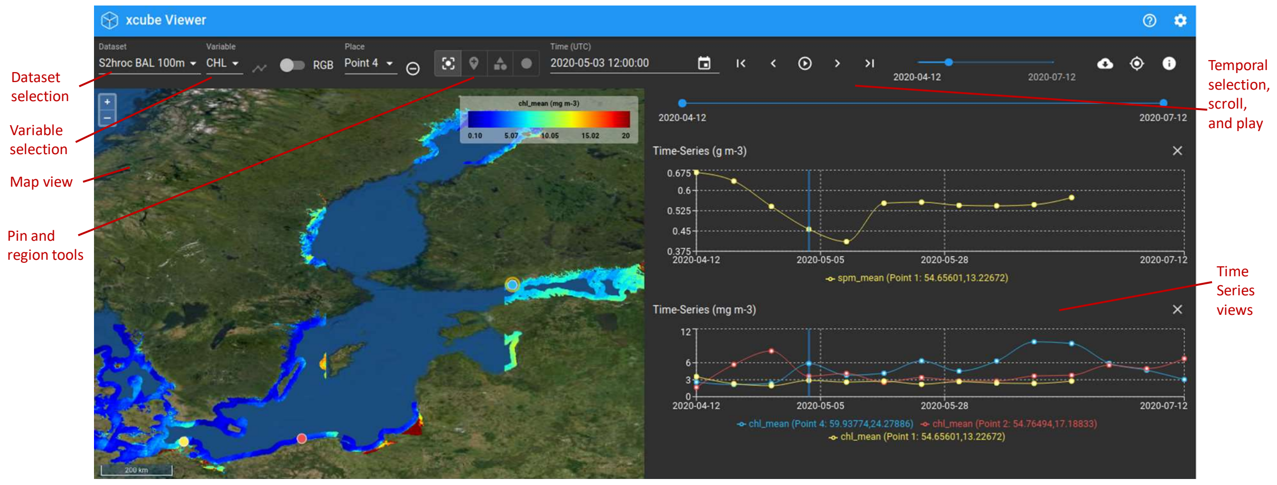


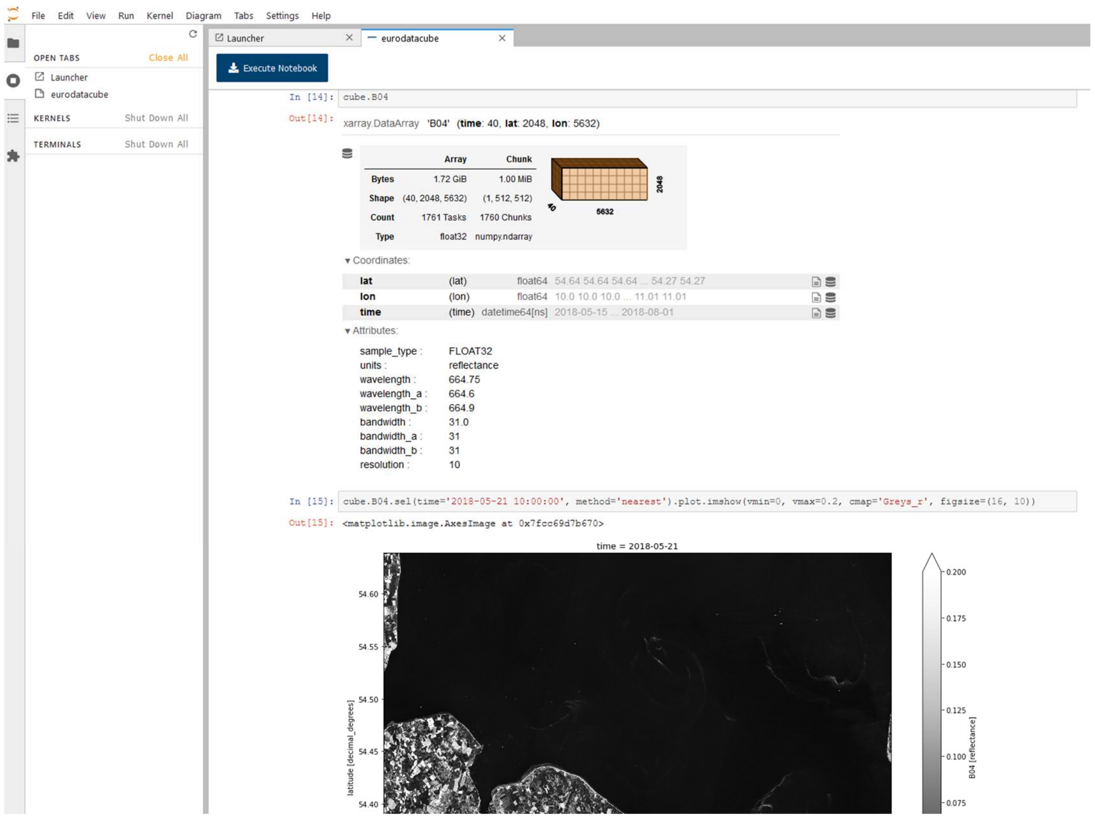
Figure 5-3: XCube Web viewer for browsing raster datasets and their time series

Users can select different datasets and variables, visualise their content, and decide which datasets to use for a showcase. Finally, users may want to integrate the dataset into their analysis using WCS and their own GIS application as described above.

5.4 Use of Jupyter notebooks for interactive analysis

BalticAIMS Jupyter notebooks are hosted in a JupyterLab server in the CreoDIAS environment. The notebooks have local access to the data using the APIs of XCube and GeoDB. This access is for expert users that use Python to adapt or implement their data analysis.

Figure 5-4 shows a Jupyter notebook with access to data cubes using the xcube API.



```
In [14]: cube.B04
Out[14]: xarray.DataArray 'B04' (time: 40, lat: 2048, lon: 5632)
```

	Array	Chunk
Bytes	1.72 GiB	1.00 MiB
Shape	(40, 2048, 5632)	(1, 512, 512)
Count	1761 Tasks	1760 Chunks
Type	float32	numpy.ndarray

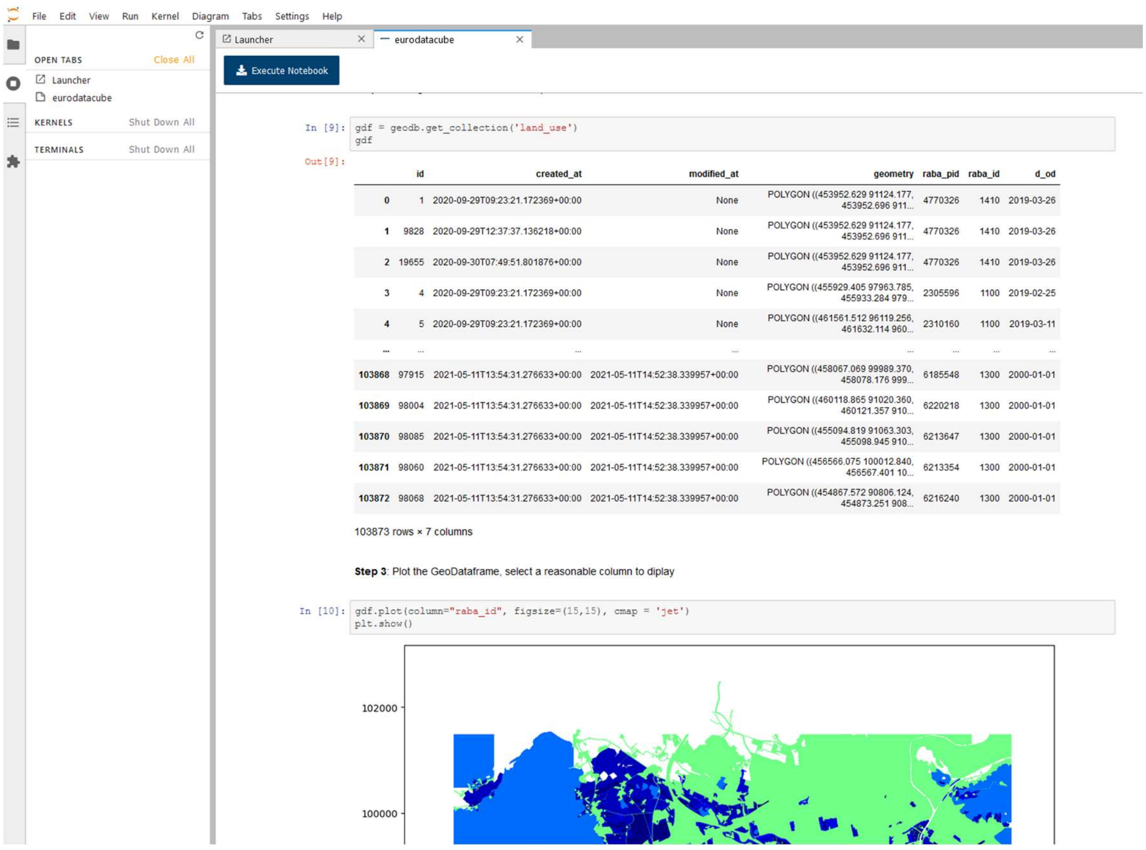
```
▾ Coordinates:
  lat     (lat)   float64         54.64 54.64 54.64 ... 54.27 54.27
  lon     (lon)   float64         10.0 10.0 10.0 ... 11.01 11.01
  time    (time)  datetime64[ns]  2018-05-15 ... 2018-08-01
▾ Attributes:
  sample_type :    FLOAT32
  units :          reflectance
  wavelength :     664.75
  wavelength_a :   664.6
  wavelength_b :   664.9
  bandwidth :      31.0
  bandwidth_a :    31
  bandwidth_b :    31
  resolution :     10

In [15]: cube.B04.sel(time='2018-05-21 10:00:00', method='nearest').plot.imshow(vmin=0, vmax=0.2, cmap='Greys_r', figsize=(16, 10))
Out[15]: <matplotlib.image.AxesImage at 0x7fcc69d7b670>
```

Figure 5-4: Jupyter notebook access to xcube, shown here for EuroDataCube as example

Using the API users

- open a connection to an xcube server
- list the available datasets and their variables
- work with the data, display it and extract time series
- combine data and generate new results

Figure 5-5 shows a Jupyter notebook with access to feature datesets using the geodb API.



```
In [9]: gdf = geodb.get_collection('land_use')
        gdf
Out[9]:
```

	id	created_at	modified_at	geometry	raba_pid	raba_id	d_od
0	1	2020-09-29T09:23:21.172369+00:00	None	POLYGON ((453952.629 91124.177, 453952.696 911...	4770326	1410	2019-03-26
1	9828	2020-09-29T12:37:37.136218+00:00	None	POLYGON ((453952.629 91124.177, 453952.696 911...	4770326	1410	2019-03-26
2	19655	2020-09-30T07:49:51.801876+00:00	None	POLYGON ((453952.629 91124.177, 453952.696 911...	4770326	1410	2019-03-26
3	4	2020-09-29T09:23:21.172369+00:00	None	POLYGON ((455929.405 97963.785, 455933.284 979...	2305596	1100	2019-02-25
4	5	2020-09-29T09:23:21.172369+00:00	None	POLYGON ((461561.512 96119.256, 461632.114 960...	2310160	1100	2019-03-11
...	...	...	...	...	...	...	...
103868	97915	2021-05-11T13:54:31.276633+00:00	2021-05-11T14:52:38.339957+00:00	POLYGON ((458067.069 99989.370, 458078.176 999...	6185548	1300	2000-01-01
103869	98004	2021-05-11T13:54:31.276633+00:00	2021-05-11T14:52:38.339957+00:00	POLYGON ((460118.865 91020.360, 460121.357 910...	6220218	1300	2000-01-01
103870	98085	2021-05-11T13:54:31.276633+00:00	2021-05-11T14:52:38.339957+00:00	POLYGON ((455094.819 91063.303, 455098.945 910...	6213647	1300	2000-01-01
103871	98060	2021-05-11T13:54:31.276633+00:00	2021-05-11T14:52:38.339957+00:00	POLYGON ((456566.075 100012.840, 456567.401 10...	6213354	1300	2000-01-01
103872	98068	2021-05-11T13:54:31.276633+00:00	2021-05-11T14:52:38.339957+00:00	POLYGON ((454867.572 90806.124, 454873.251 908...	6216240	1300	2000-01-01

103873 rows × 7 columns

Step 3: Plot the GeoDataframe, select a reasonable column to diplay

```
In [10]: gdf.plot(column="raba_id", figsize=(15,15), cmap = 'jet')
         plt.show()
```

Figure 5-5: Jupyter notebook access to geodb, shown here for EuroDataCube as example

Using the API users

- open a connection to a geodb server
- list the available datasets and their attributes
- filter the data
- plot the results

With Jupyter notebooks users can apply their own methods to the data and to process it to get specific results for their showcase.

6 References

[D2.1]
D2.1 Service Portfolio Definition

[D2.2]
D2.2 Data and Platform Provisioning Plan

[A2.1 Datasets Table]
BalticAIMS Dataset Attributes tables, Excel tables, version 2.0 (in progress, retrieved 16.09.2021)