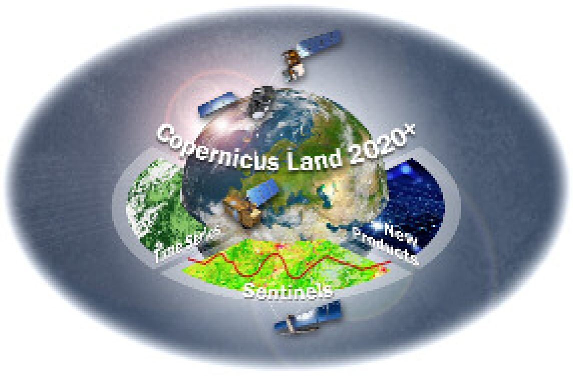
Horizon 2020

Space Call - Earth Observation: EO-3-2016: Evolution of Copernicus services

Grant Agreement No. 730008

ECoLaSS

Evolution of Copernicus Land Services based on Sentinel data



D5.1

"D23.1: Service Infrastructure/Architecture Requirements Report"

Issue/Rev.: 1.1

Date Issued: 20.12.2017

submitted by:

DLR

in collaboration with the consortium partners:

SIRS JOANNEUM RESEARCH DIGITAL UCL Université catholique de Louvain GAF AG

submitted to:

European Commission – Research Executive Agency

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No. 730008.

Consortium Partners

No.	Participant Organisation Name	Short Name	City, Country
1	GAF AG	GAF	Munich, Germany
2	Systèmes d'Information à Référence Spatiale SAS	SIRS	Villeneuve d'Ascq, France
3	JOANNEUM RESEARCH Forschungsgesellschaft mbH	JR	Graz, Austria
4	Université catholique de Louvain, Earth and Life Institute (ELI)	UCL	Louvain-la-Neuve, Belgium
5	German Aerospace Center (DLR), German Remote Sensing Data Center (DFD), Wessling	DLR	Wessling, Germany

Contact:

GAF AG
Arnulfstr. 199 – D-80634 München – Germany
Phone: ++49 (0)89 121528 0 – FAX: ++49 (0)89 121528 79
E-mail: copernicus@gaf.de – Internet: www.gaf.de

Disclaimer:

Document Release Sheet

	Name, Function	Date	Signature
Author(s):	Andreas Hirner (DLR) Gernot Ramminger (GAF) Sophie Villerot (SIRS)	12.12.2017	[illegible]
Review:	Annekatrin Metz-Marconcini (DLR) Linda Moser (GAF)	19.12.2017	[illegible]
Approval:	Linda Moser (GAF)	20.12.2017	[illegible]
Acceptance:	Massimo Ciscato (REA)		
Distribution:	Public		

Dissemination Level

Dissemination Level		
PU	Public	X
CO	Confidential: only for members of the consortium (including the Commission Services)	

Document Status Sheet

Issue/Rev	Date	Page(s)	Description / Changes
1.1	20.12.2017	24	First issue of the Service Infrastructure/Architecture Requirements Report

Applicable Documents

ID	Document Name / Issue Date
AD01	Horizon 2020 Work Programme 2016 – 2017, 5 iii. Leadership in Enabling and Industrial Technologies – Space. Call: EO-3-2016: Evolution of Copernicus services. Issued: 13.10.2015
AD02	Guidance Document: Research Needs Of Copernicus Operational Services. Final Version issued: 30.10.2015
AD03	Proposal: Evolution of Copernicus Land Services based on Sentinel data. Proposal acronym: ECoLaSS, Proposal number: 730008. Submitted: 03.03.2016
AD04	Grant Agreement – ECoLaSS. Grant Agreement number: 730008 – ECoLaSS – H2020-EO-2016/H2020-EO-2016, Issued: 18.10.2016

EXECUTIVE SUMMARY

The Horizon 2020 (H2020) project, "Evolution of Copernicus Land Services based on Sentinel data" (ECoLaSS) addresses the H2020 Work Programme 5 iii. Leadership in Enabling and Industrial technologies - Space, specifically the Topic EO-3-2016: Evolution of Copernicus services. ECoLaSS will be conducted from 2017–2019 and aims at developing and prototypically demonstrating selected innovative products and methods for future next-generation operational Copernicus Land Monitoring Service (CLMS) products of the pan-European and Global Components. This will contribute to demonstrating operational readiness of the finally selected products, and shall allow the key CLMS stakeholders (i.e. mainly the Entrusted European Entities (EEE) EEA and JRC) to take informed decisions on potential procurement of the next generation of Copernicus Land services from 2020 onwards.

To achieve this goal, ECoLaSS will make full use of dense time series of Sentinel-2 and Sentinel-3 optical data as well as Sentinel-1 Synthetic Aperture Radar (SAR) data. Rapidly evolving scientific as well as user requirements will be analysed in support of a future pan-European roll-out of new/improved CLMS products, and the transfer to global applications.

The Horizon 2020 (H2020) project, "Evolution of Copernicus Land Services based on Sentinel data" (ECoLaSS) comprises several EO institutions and companies with different IT infrastructures. These infrastructures developed in accordance to the processing requirements faced by each stakeholder and were increasingly influenced by three general trends that developed during the last 10 years. These trends are:

- Availability of medium to high resolution EO data with a high temporal resolution on a global scale
- the possibility to create cheap off-the-shelf computer clusters that can be adapted for a wide range of storage and processing tasks
- a big set of (mostly Open Source) processing frameworks that monitor the hardware and help organize the processing tasks

In order to cope with different setups of single, independent IT structures several techniques show great promise for interaction:

- Virtualization allows decoupling of processing algorithms from the underlying hardware or OS and therefore, minimizes dependency management
- Processing services offer the possibility to run anybody's algorithm on huge data sets by exchanging code instead of data

Adoption of the last two strategies should enable increased collaboration of project partners and help making the transition from the test to the production processing environments much easier.

In order to be able to assess future service infrastructure architectures, an evaluation catalogue is presented in this report, which will help to analyse the capabilities of potential processing platforms accessible to the project. The result of this evaluation will be reflected in the second issue of this deliverable. However, the catalogue will also be helpful in choosing an appropriate architecture of the tools provided by this project.

This document constitutes an updated issue 1.1 of the Service Infrastructure Requirement Report (D23.1a), broadening the thematic content as requested in the Interim Progress Meeting 1.

Table of Contents

List of Figures and Tables

Abbreviations

Apache SF	Apache Software Foundation
API	Application Programming Interface
AWS	Amazon Web Services
CPU	Central Processing Unit
DAGs	Directed Acyclic Graphs
DEM	Digital Elevation Model
DIAS	(Copernicus) Data and Information Access Services
EC	European Commission
EEA	European Environment Agency
EO	Earth Observations
EOC	Earth observation centre
ESA	European Space Agency
GB	GigaByte
GDAL	Geospatial Data Abstraction Library
GEE	Google Earth Engine
GFS	Google File System
GPFS	General Parallel File System
GPU	Graphical Processing Unit
HDFS	Hadoop Distributed Filesystem
HPC	High Performance Computing
HR	High Resolution
HRL	High Resolution Layer
I/O	Input/Ouput
ICT	Information and Communications Technology
IT	Information Technology
NAS	Network Attached Storage
OGC	Open GIS Consortium
OS	Operating System
OTB	Orféo ToolBox
RAM	Random Access Memory
RDD	Resilient Distributed Dataset
REST	Representational State Transfer
RUS	Research and User Support
SF	Software Foundation
SSD	Solid State Discs
SATA	Serial AT Attachment
TB	TeraByte
TEP	Thematic Exploitation Platform
ToA	Top of Atmosphere
USGS	United States Geological Survey
WBS	Work Breakdown Structure
WMS	Web Mapping Service (OGC)
WFS	Web Feature Service (OGC)
WP	Work Package
WPS	Web Processing Service (OGC)
WPD	Work Package Description
XML	Extensible Markup Language
YARN	Yet Another Resource Negotiator
ZFS	Zetta File System

For more specific terms see chapter 1.3. – Terminology.

1 Introduction

1.1 Future Challenges

The launch and operation of the Sentinel 1, 2 and 3 Earth observation satellites has resulted in a sharp increase of remote sensing data (SERCO Spa, 2016) for global land and ocean monitoring services. Together with already existing data sets (e.g. Landsat) users suddenly see new possibilities and are willing to tackle questions with a global scope, which is not limited to a single time slice but tries to reveal changes and developments in order to explain the dynamics of land and ocean processes.

Rapid developments in the IT industry epitomized by the buzz words of “big data” and “cloud” open new ways of dealing with large volumes of data. Large data volumes do not need to be downloaded and stored in local computing environments, now they can be processed on site. New processing paradigms change the way data can be processed and IT provider offer different environments, which are geared towards specific tasks.

Therefore, users face the problem of dealing with massive EO data sets in a new way. Approaches taken by different organizations are highly diverse and rapidly evolving, especially since the appearance of readily available commercial EO processing platforms (e.g. Google Earth Engine, Amazon Web Services).

1.2 Components

Hardware of a minimal processing system usually consists of at least the following components:

- Storage units
- Processing units

In its simplest form, this can be a single computer, but with large volumes of data and many processing tasks, storage is distributed over many hard drives and computing is done by multiple processors with many cores spread across many computers. Therefore

- Network infrastructure

becomes an important issue. There are many ways to organize computer infrastructure, which strongly depend on the main purpose of the processing system. A subsequent chapter will discuss the rationale and consequences of different hardware setups. In addition to that, fault tolerance requires dedicated nodes for monitoring and managing the hardware. The same is valid for scheduling and managing the processing tasks, resulting in hardware solely responsible for process execution, coordination and orchestration.

Minimal processing software might be just one or more executables that are triggered on demand, but growing hardware complexity also leads to specialised software components. Common software components are used for:

- Data transformation
- Data transfer
- (Scientific) computing
- Task scheduling
- Task monitoring
- Hardware resource monitoring

These functions are also dependent on the processing infrastructure and can get extremely complex, requiring dedicated software packages for each of them.

1.3 Terminology

Many terms in computing are ambiguous and may have a different meaning in a given IT subdomain. This chapter describes terms as they are considered in the frame of this document.

Resource	a storage (usually measured in Bytes) or processing unit (usually measured in CPUs or cores)
Cloud	the sum of specific computing infrastructures with specific interfaces accessible in the internet (public cloud) or intranet (private cloud). Computing infrastructures usually offer specific services that can be categorized in different categories
IaaS (Infrastructure as a service)	offers computing resources in the form of operating systems, virtual machines, container or storage
PaaS (Platform as a service)	offers services to develop and host custom built applications
SaaS (Software as a service)	offers software like databases, servers, default applications, etc.
Orchestration	a procedure that accepts job requests, monitors available resources and triggers new jobs whenever there are sufficient resources
Virtual machine	an hardware system virtualization managed by a hypervisor software running on a host
Container	an operating system level virtualization managed by the kernel of a host OS
I/O operations	read and write operations
Node	one computer in a computer cluster

2 Analysis of service infrastructure and architecture

2.1 Processing concepts and challenges

Each setup of a processing system has advantages and disadvantages. This chapter will highlight common aspects regarding storage, networking and processing units.

Storage can be organized in several ways.

- A data storage server with many hard drives connected to a network (e.g. NAS – network attached storage)
- A group of ordinary computers with one more hard drives

In the first case, the storage server is the single point of entry and might become a bottleneck once the data flow through the network gets saturated. A big advantage of this setup is its ability to hide the single hard drives and offer the sum of their total volume as a single logical unit and provide redundancy.

The latter case allows linking cheap off the shelf computers in a network in such a way that each can be accessed via the network. This increases the potential data transfer volume, but has one big drawback: How are data distributed and accessed? In a primitive setup data must transferred manually and if storage of a given computer is full, data relocation and even modification will become a problem. The same applies for search and access functions. Each computer must be searched and indexed in order to find a data set. Despite that this setting offers a big advantage regarding processing: Data on a given computer can be processed with the CPUs of this machine without data transfer across the network. This opens the possibility for parallel processing. But if it is cumbersome to locate data, how are processing tasks assigned according to different search criteria? There are numerous strategies to solve these problems and developments in the last decade addressed many problems and provided technical solutions. They will be discussed in the following chapter.

2.2 Analysis of current situation and limitations

One development regards distributed file systems (also called network file systems). Here multiple hard drives across many computers are linked by a specific protocol and managed by dedicated software present on each node. There are numerous implementations with different design goals like transparency (clients reading data do not need to know where data are), scalability, redundancy and heterogeneity. Common examples are *General Parallel File System* (GPFS) by IBM (IBM, n.d.), *Google File System* (GFS) by Google (Ghemawat, Gobioff, & S.T., 2003), *Zetta File System* (ZFS) by Oracle (Oracle, 2012), *Hadoop Distributed Filesystem* (HDFS) by Apache SF (Apache, 2013) and many more.

Another development is concerned with computing resource management. While distributed file systems allow users to manage disc space, the load of CPUs on each node has to be monitored in order to effectively distribute processing tasks. The main components of cluster management software are job scheduler and workload manager. The former accepts processing requests and stores them in a queue, the latter monitors the load of computer nodes. Whenever there are computing resources available on one of the nodes, the workload manager searches the schedulers for a matching task and assigns the task to that node for execution.

Processing requests often have to be registered beforehand with the job scheduler in order to link the processing software with required hardware resources like RAM and number of cores. This is important in order to optimize compute resource allocation. In addition to that cluster management software includes job monitors and might offer tools for deployment, scaling and other tasks. Popular examples are YARN (Yet Another Resource Negotiator) and Mesos (Apache, 2017), both by Apache SF, and Kubernetes (Linux Foundation, 2017) by Google.

An area linked to cluster management deals with the combination of single jobs into complex workflows. This is sometimes also part of orchestration. Dedicated software systems allow users to define tasks, construct workflows and register processing endpoints on local or public processing systems. Once triggered the orchestration software conditionally executes a workflow and logs its progress and status. Popular products are Ansible (Red Hat, Inc., 2017) developed by Red Hat, Jenkins (Jenkins Project, n.d.) by Open Project, Airflow (Apache Inc., n.d.) and Mesos (both developed by Apache SF).

Substantial improvements regarding the configuration and deployment of processing software was achieved with containers. Containers are an operating-system-level virtualization technique, which allows the execution of an encapsulated OS on a host computer. Since the technique only requires very limited resources with a minimal footprint it is possible to run many containers on a host system. The main strength of this technique is the fact that a user can construct a container image that contains all processing software needed for a specific task. Once the container image is constructed, it can be deployed on the nodes of a processing cluster. If deployment is complete a job request can create a container and execute the software provided by it. If the job is complete the container is destroyed and its resources are freed. Using containers removes the need to install specific software packages on the processing nodes and considerably simplifies the management dependencies. Well known examples are Docker (Docker Inc., 2017) and LXC (Canonical Ltd, n.d.) .

Last but not least there are many efforts to increase processing algorithms itself. A major processing bottleneck is I/O operations. The development of stream processing frameworks tries to limit read and write operations and provides software tools that allow feeding datasets through a system of pipes where the data stream can be modified or analysed on the fly before the final product is written to the disc. Other processing frameworks are adapted to the structure of a specific data type and provide optimised tools and procedures for their efficient handling. They will be introduced in the following chapter.

A final paragraph is concerned with hardware development. The power of graphical processing units (GPUs) has increased massively due to the ever-rising demands of the gaming industry. But this kind of hardware is also perfectly suited for massively parallel processing tasks. This capability is now exploited in numerous specific fields (e.g. machine learning, geometric calculations). Developments of solid-state drives (SSD) opened up new possibilities regarding storage. I/O operations on these types of disk are much quicker and can lead to significant performance boosts during processing. Creation of specific computer systems with large amounts of RAM is also geared towards the speed-up of I/O operations. All these techniques are often used together with conventional CPUs and discs in a hybrid setup.

It should be mentioned that large scale processing has traditionally been done on high performance computing (HPC) clusters with specific hardware geared towards optimal performance. These systems are usually very expensive and the access is commonly restricted. However, there are increasing numbers of HPC services from public and commercial providers, which can be utilized for specific tasks.

2.3 Processing paradigms: background and implications

The chapter above was concerned with developments in separate fields, but recent trends have resulted in processing paradigms that describe recipes for specific processing scenarios. These recipes were picked up by the Open Source community in concert with commercial actors, which implemented processing frameworks that can be adapted and utilized by anybody.

An influential development was the Map-Reduce concept published by Google in a scientific paper in 2003 (Ghemawat, Gobioff, & S.T., 2003). The concept describes how large data sets are broken up and distributed across processing nodes where each fragment is analysed separately (Map). The results from each node are than collected and aggregated (Reduce).

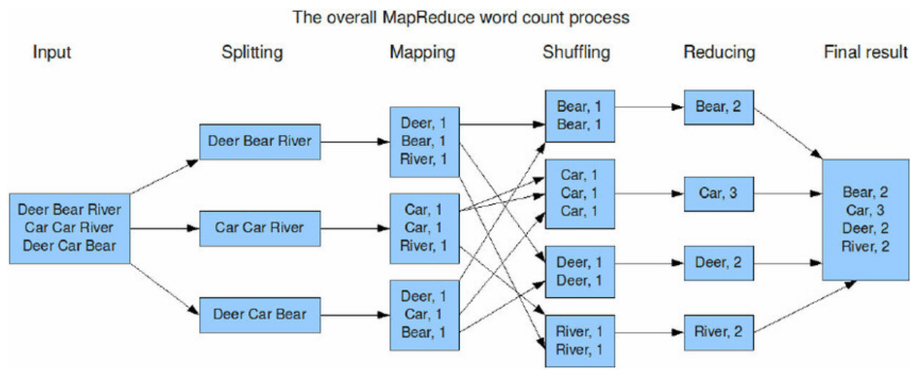


Figure 1: MapReduce data flow

The idea was immediately picked up by developers and resulted in the Apache-Hadoop project. The central components of Hadoop are the Hadoop Distributed File System (HDFS) and Hadoop YARN, a platform responsible for managing computing resources in clusters. A third component is the Hadoop MapReduce API, which defines an interface for developers to implement their own tasks. The Hadoop framework has since grown into large ecosystem, where HDFS and Yarn can be used to manage not only Map-Reduce tasks but also other processing paradigms.

One of those is implemented in the Apache Spark framework, which permits in-memory querying of data instead of disk I/O operations. Spark relies on its own data format called resilient distributed dataset (RDD) and is thus able to implement iterative algorithms that visit their dataset multiple times in a loop, and analysis like the repeated database-style querying of data. Therefore, it is better suited for tasks where data are related to each other. It also allows constructing workflows as multi-step directed acyclic graphs (DAGs) (Thulasiraman & Swamy, 1992) and executing those DAGs all at once, not step by step.

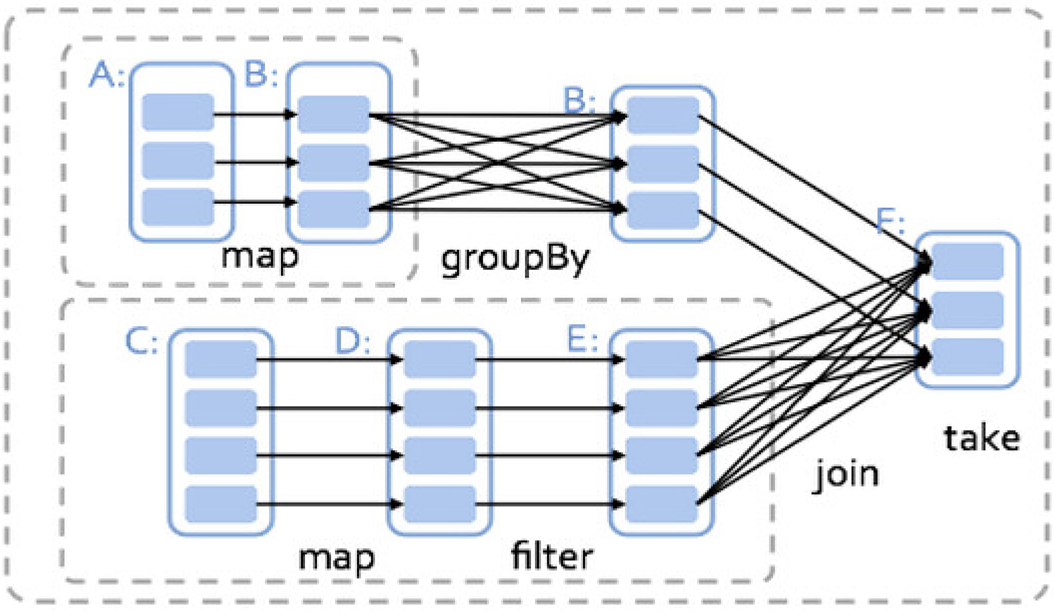


Figure 2: Data flow in Apache Spark

The concept of DAGs is also used by other frameworks that are used construct complex workflows, e.g Ansible and Apache Airflow.

Other frameworks (Apache Storm, Apache Kafka, Apache Flink) read data and create a stream that can be manipulated in successive steps. These systems achieve an extremely high transfer volume and are able to react on events and define workflows in the form of DAGs. Some of the systems are able to split and distribute the streams to different nodes and therefore increase processing speed. Others read data from a range of producers and allow consumers to query and analyse one or more of them.

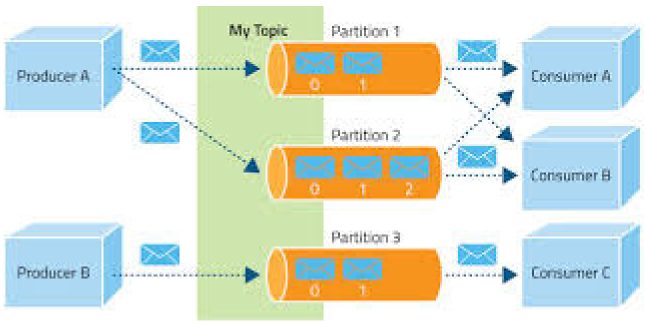


Figure 3: Data flow in Apache Kafka

A strictly organisational approach was pursued by the Apache Mesos framework. Mesos provides applications with APIs for resource management and scheduling across entire data centres and cloud environments. It can orchestrate many different processing frameworks (e.g., Hadoop, Spark, Kafka, Elasticsearch) and most importantly, supports launching Docker containers. It is highly fault tolerant and scalable. Main components are the Chronos scheduler and the Marathon orchestration system.

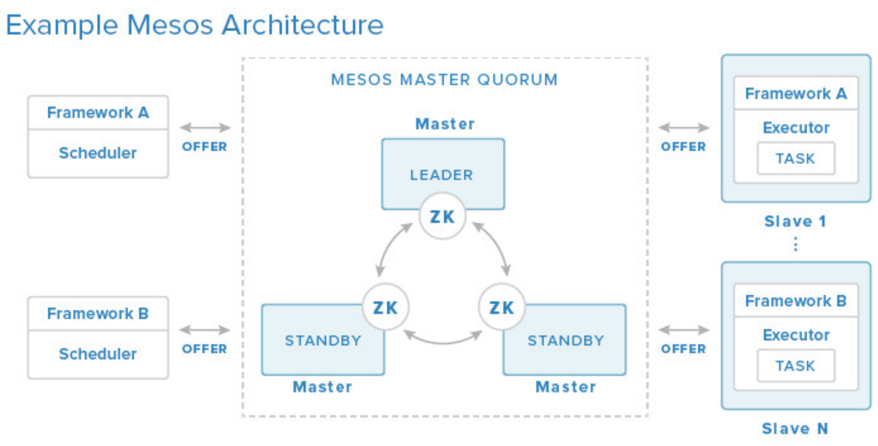


Figure 4: Mesos concept

Similar systems geared strictly towards containers are Kubernetes (Google) and Docker Swarm (Docker Inc.). They share many of the same goals of Mesos, but also manage the distribution and provisioning of containers across nodes.

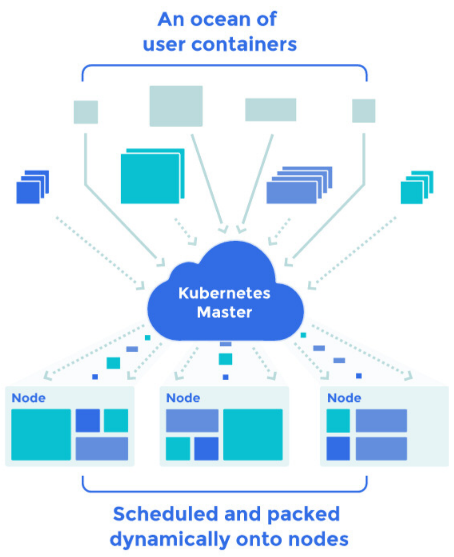

An ocean of user containers
Kubernetes Master
Node
Node
Node
Scheduled and packed dynamically onto nodes

Figure 5: Kubernetes concept

The specific requirements of EO data processing have already attracted service providers to offer specific solutions by Google and Amazon. Both services run on the server farms of the two companies and are accessible in the cloud.

Google Earth Engine, abbreviated intoGEE (Google, 2017), computing platform provides a large collection of satellite data (e.g. Landsat, MODIS, Sentinel-1 and -2, and DEMs) that are registered in a catalogue and a corresponding set of APIs for Python and JavaScript for statistical and geospatial analysis. Customers use the API to write their own code, which selects the desired data and algorithms and subsequently triggers processing. The size of the results usually is much smaller than the amount of processed data and can be down loaded by the user. In addition, a user can upload ancillary raster and vector data. Upon registration, the offer is free for research, education and non-profit usage.

Amazon also offers a range of geospatial and satellite data in their S3 storage engine (Amazon Web Services, Inc., 2017). These include Landsat, Sentinel 2, MODIS, Goes, Nexrad, DEM and OSM. A user can work with the tools of Amazon Web Services (AWS) in order to write code, which accesses and manipulates data in a similar way as GEE. The range of tools is extensive (including a range of languages, APIs and tools like databases, webservers, etc.) and there are more different ways to find a solution compared with GEE. The services are billed according to the AWS business model and include costs for data storage, processing and transfer.

SAP using its HANA in-memory database also offers services for processing geospatial data (SAP, 2017). The technique allows different modes of storing and processing data with a minimal amount of I/O operations. The technique shares many of the capabilities of Apache Spark.

2.4 EO data initiatives in Europe

In Europe, the Copernicus initiative of EU and ESA has resulted in many national and regional initiatives to provide and handle Sentinel data sets. Collaborations between commercial service providers, universities and research institutes are currently being planned and implemented. Examples (ESA, 2017) include, among others:

- the Code-DE project in Germany (Infrastruktur, BVDI - Bundesministerium für Verkehr und digitale, 2017),
- EODC in Austria (EODC Earth Observation Data Centre for Water Resources Monitoring GmbH, 2017),
- PEPS in France (CNES - Centre National d'Études Spatiales, 2017),
- Terrascope for Belgium (VITO - Vlaamse Instelling voor Technologisch Onderzoek, 2017),
- the Hellenic National Sentinel Data Hub for Greece (NOA - National Observatory of Athens, 2016)
- the upcoming DIAS by the EC.

They are all designed for processing EO data and adhere, with some deviations, to one of designs mentioned above.

2.4.1 Thematic Exploitation Platform (TEP)

There are various programs by EC and ESA to explore the potential utilization of EO data by a large community of users.

An example is the Thematic Exploitation Platform (TEP) initiative by ESA (ESA, 2017, a). Here users of various topics (e.g. Coastal, Forestry, Hydrology, Geohazards, Polar, Urban themes; and Food Security) are encouraged to use an on-line processing platform that provides tools and relevant data to evaluate and analyse certain aspects of their research field. This is in stark contrast to the previous practice of downloading tools and data to a local processing destination, which can be quite time consuming and ineffective. In addition, this approach allows the provision of restricted data that can be visualized and even used in calculations. In the long run, each platform is seen as a common play ground that fosters rapid development of ideas, products, tools and even commercial activities.

Each exploitation platform has to solve similar problems regarding storage, processing and user interaction. The solutions developed in each project will come with specific conventions, services and interfaces bundled into a web application and corresponding back-end systems. The lessons learned from these systems can betray potential preferences and concepts, which might lead to general recommendations and therefore improve the design of other projects.

2.4.2 Research and User Support (RUS) Service portal

The RUS service (ESA, 2017, b) is an expert service for Sentinel users funded, managed and operated by the European Commission, European Space Agency and CS SI respectively. Here registered users can obtain a personal computing environment, which includes a virtual desktop environment based on Linux and connected to various cloud services to provide the user with the required resources (CPU, memory, storage, etc.). The user is able to utilize and enhance his computing environment like an ordinary Linux system, by adding the necessary libraries for his development tasks. In addition to that the user can connect to his environment with other means like ssh/pssh (interaction) and ftp (data exchange). Tools like scihub_download enable quick access to Sentinel data. It is therefore possible to rapidly setup a processing environment and work with Sentinel data. Platforms like that might serve as a test ground for products developed within the consortium.

2.4.3 Data and Information Access Service (DIAS)

The Copernicus Data and Information Access Service (DIAS) aims to maximize the exploitation of Copernicus data by a broad user community adopting a dynamic approach using the latest ICT and EO technologies (European Commission, 2017). It will focus on increased industry responsibility in a federation of centres that consist of distributed and integrated ground segments.

Each of the centres offers so called operations that fall into two categories, the Back-office and Front-office operations. A given DIAS provider will be in charge of the entire storage and processing infrastructure, its services and corresponding interfaces, which make up a single Back-office. These services shall enable efficient local processing of the complete local archive of Copernicus data and/or existing Copernicus distribution services. A Back-office is supposed to be scalable and will include a broad range of services, interfaces and tools to access and manipulate the data. In addition to data retrieval and manipulation modules, users can also expect discovery, catalogue and view services.

Products generated and services provided by the Back-office operations shall be utilized by Front-office operations. Any third party that wishes to exploit the data can use the Back-office infrastructure to generate their own products and related services in a Front-office entity. Therefore, the DIAS provider will offer technical support for integration of third party data sources and software whenever necessary. The concept also envisions that services and products of any third party is also usable by other third parties, thus establishing a broad utilization of data while at the same time trying to minimize redundancies.

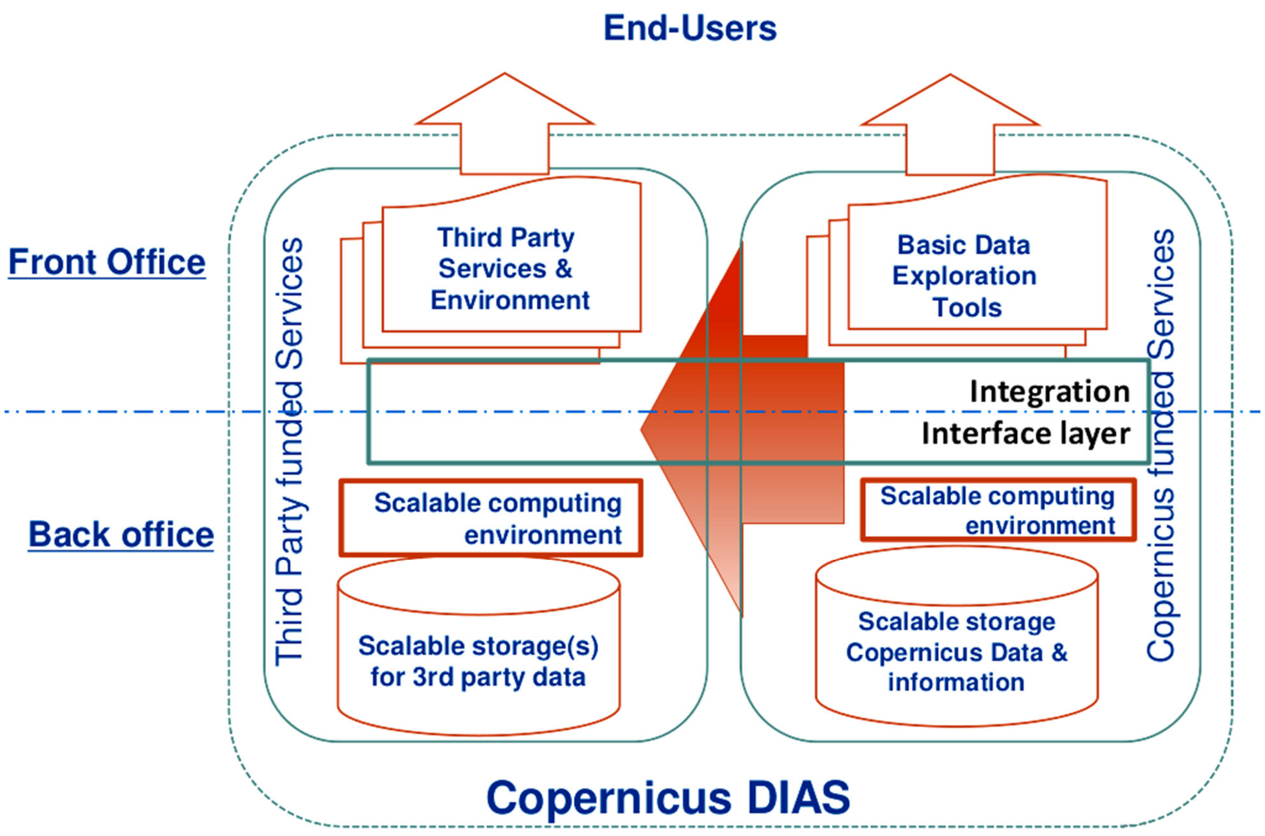

End-Users
Front Office
Back office
Third Party funded Services
Third Party Services & Environment
Basic Data Exploration Tools
Integration Interface layer
Scalable computing environment
Scalable storage(s) for 3rd party data
Scalable computing environment
Scalable storage Copernicus Data & information
Copernicus funded Services
Copernicus DIAS

Figure 6: Dias concept

Due to this construct a vital part of DIAS will be the quality, stability but also the flexibility of interfaces and services. Are they to rigid, users will find the system inconvenient to utilize and move to other service providers. If they are too instable, a third-party Front-office user will consider the constant effort for maintenance to tiresome and also look for another provider. It should therefore aim to closely resemble established and thus stable industry and consortia standards, which will help users to quickly adapt to the platform without major adjustments. It will also be necessary to offer not just one processing paradigm, but maybe several different ones in order to offer the right tools for a given processing task.

Apart from the software capabilities the underlying hardware platform needs to be powerful, scalable and adaptable while still being cost efficient. Unfortunately, no detailed description of any DIAS components is not yet published, but the evaluation guide line established below will applied as soon as the details of the components become available.

On 14th of December DC GROW announced consortia that will implement and run the five DIAS centres (European Commision, 2017):

- EUMETSAT, in cooperation with ECMWF and Mercator Ocean
- SERCO with OVH as cloud provider
- CREOTECH INSTRUMENTS with CLOUDFERRO as cloud provider
- ATOS INTEGRATION with T-SYSTEM INTERNATIONAL as cloud provider
- AIRBUS DEFENCE AND SPACE with ORANGE as cloud provider.

The consortium expects that DIAS will be a primary target platform for products created in the frame of this project and hopes that the already existing collaboration of SIRS/OVH and DLR/T-Systems in various projects will help to rapidly tailor our software and data procedures to the DIAS infrastructure.

2.5 Processing in the consortium

2.5.1 High Resolution Layer production - Experiences from GAF

In June 2016, GAF together with its partners SIRS, GeoVille and e-Geos were awarded by EEA with the production of the five High Resolution Layers 2015. The following section provides an overview about the chosen Infrastructure and Architecture, required for the production of the HRLs Update 2015, where GAF was responsible for. The HRLs are produced mainly on 20m spatial resolution for an area of approx. 5.850.000 km² (EEA39) and covering multiple points in time:

- Imperviousness Layer: Built-up area & imperviousness degrees 2015, re-processing of 2006-2009-2012 time series, imperviousness change and reference database,
- Forest Layer: Dominant leaf type 2015 and change to 2012, tree cover density 2015 and change to 2012, forest reference database,
- Permanent grassland mask 2015, grass vegetation probability index 2015, ploughing indicator (back to 2008),
- Water & wetness 2015, water wetness probability Index 2009 to 2015,
- Small woody features 2015 based on VHR/HR data 2015 (+/- 1 year)

2.5.1.1 Requirements on the selected Service Infrastructure and Architecture of GAF

For the production of the HR Layers, GAF was mainly (beside other) responsible for the following tasks also providing the main requirements on the service infrastructure and architecture:

- pre-processing of all optical data (timeframe from 2005 to 2016, covering 8 different sensors from HR IMAGE2006 and HR IMAGE2009 (IRS-P6, Spot-4, Spot-5), HR IMAGE2012 (IRS-P6, Resorcesat-2, Spot-4, Spot-5) HR IMAGE2015 (Resourcesat-2, Spot-5, Sentinel-2) as well as additional Sentinel-2 and Landsat-5/-7/-8 (2005 to 2016)
    - o data selection and ordering system
    - o centralised storing/archiving system (for ~ 70 TB data or ~ 65.000 datasets)
    - o automated, centralised, pre-processing system with parallel processing capabilities to handle data orders of up to 2000 scenes in parallel (cloud-/cloud-shadow masking, ToA

correction, geometric validation, geometric correction (to a Sentinel-2 Level 1C geometric reference mosaic), topographic normalisation and indices calculation,)
    - dissemination system to provide pre-processed data to all partners
- production of the forest (~5.850.000 km²) and grassland layers (~1.930.500 km²)
    - mainly automated centralised and database driven thematic processing system (multi-temporal satellite data)
    - parallel processing capabilities (multiple production units per week)
    - high performance computing power for image segmentation

2.5.1.2 Selected Service Infrastructure and Architecture Setup for HRL production

For the HRL production, GAF selected a hybrid setup using:

- capabilities of its own private cloud for parallel processing, storing, archiving together with
- the public cloud (AWS) especially for storage and data dissemination.

A process to the data approach, using e.g. one public cloud in a more centralised was not appropriate, as required EO data was only available on different platforms/systems (e.g. ESA Data Warehouse, Sentinel-2 Copernicus Hub, Amazon, Google and USGS). An overview about the selected approach is given in the following Figure 1.

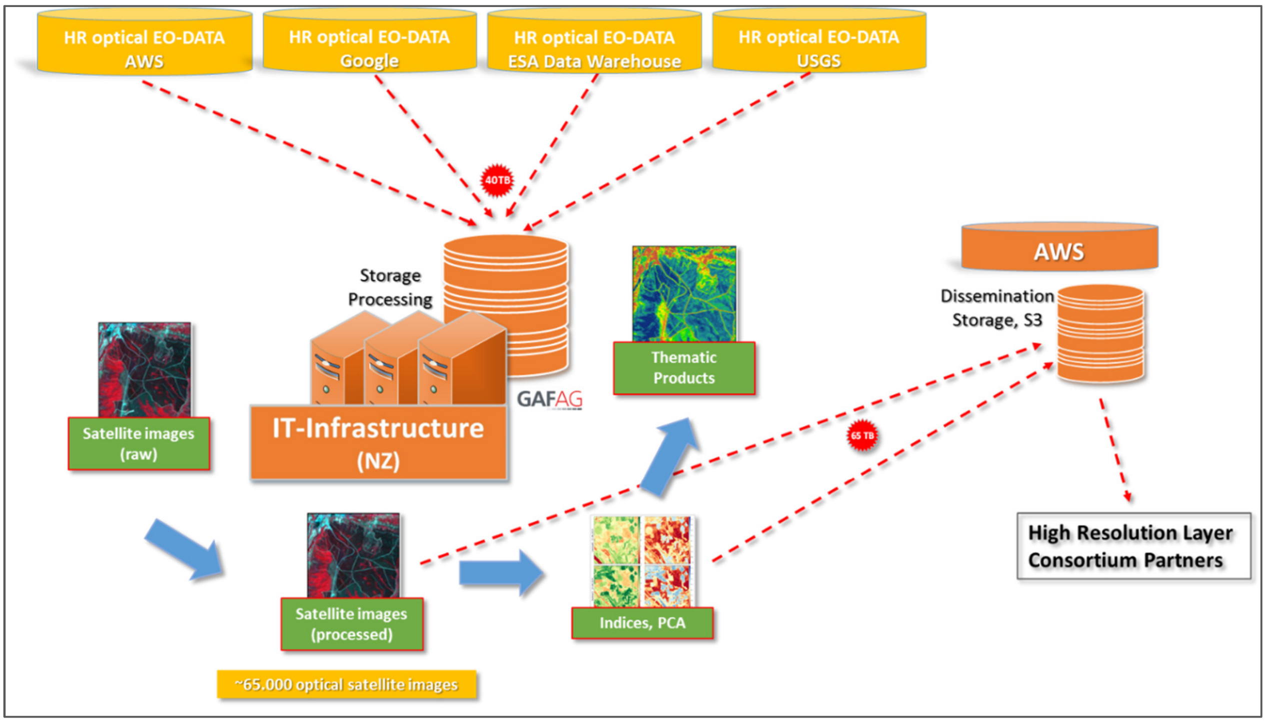


Figure 7: Overview of Service Infrastructure and Architecture Setup of GAF for the production of the HR Layers

A private cloud infrastructure, already available at GAF premise, was selected as central processing hub for GAFs activities with respect to pre-processing of optical EO data and thematic processing of the HR Forest as well as Grassland (part, covered by GAF) layers. All EO optical data selected by GAF and Consortium partners, where acquired from various sources (ESA Data Warehouse, ESA Copernicus Hub, AWS, Google and USGS) through automated download procedures and stored in a central data hub in the private cloud (80 TB on StrongBox reserved for the project, accessibility fast and medium). After metadata extraction and storage into a central database, an orchestrated (GAF intern development in C++) automated workflow to pre-process all required EO data was implemented using a scalable (up to 30 virtual processing nodes, each 16 to 32 GB RAM and 4 CPUs) processing environment based on internal GAF tools implemented in C++. Based on the set-up described before, around ~65.000 optical satellite scenes were downloaded and pre-processed during the project lifetime (with bunch of data within the first 3-4 months). Most of the pre-processed data (some data was only used by for the production of the HR Forest Layer) was uploaded afterwards to Amazon Web Services (AWS), where a centralised data hub was established. This hub on AWS served as big data dissemination hub, accessible for relevant Consortium partners with regular status updates and automated download capabilities on partner request. Most of thematic processing of the HR Forest Layers, which followed the pre-processing, was also implemented in Python using a centralised approach on the private cloud and based on up to 5 Linux server with 32 to 64 GB RAM each for automated processes as well as 9 virtual desktops for semi-automatic and/or manual processes. Some post-processing procedures were done on local workstations/desktop systems.

2.5.1.3 Lessons learnt and outlook

The development of a more centralised approach (than described before), where consequently processes (or Software environment) are moved towards the data using the same infrastructure, was not implemented as almost all of the required EO data was only available on different cloud infrastructures or different data hubs. Using a more centralised approach based on GAFs private cloud allowed scalability with respect to processing and storage capability. Nevertheless, especially data transfer (download/upload) and data storage were identified as potential bottlenecks at the beginning of the project and therefore upgraded during project lifetime. All activities described above were implemented using mainly GAFs private cloud environment (with the bottlenecks identified before) and AWS for data dissemination. The whole setup could also be transferred completely into a public cloud environment (e.g. potential cost reduction, if any). Implementation based only on traditional infrastructure e.g. single desktop systems/work stations would not be recommended anymore. A more centrally organised data access (as planned e.g. with the upcoming DIAS platforms, financed by ESA), where all free satellite systems such as the Sentinels and Landsat should be accessible on a common platform (IaaS, SaaS, PaaS) together with commercial data available from the ESA Data Warehouse, would also improve the development of a fully centralised processing approach. This approach could:

- reduce time and cost for data transfer and
- allow higher scalability of processing and storage

2.5.2 EO data processing at SIRS

SIRS mainly uses two different processing platforms. A private cloud serves as a data warehouse and provides storage for vector and alphanumeric data in a Database (PostgreSQL) and raster data on a NAS system.

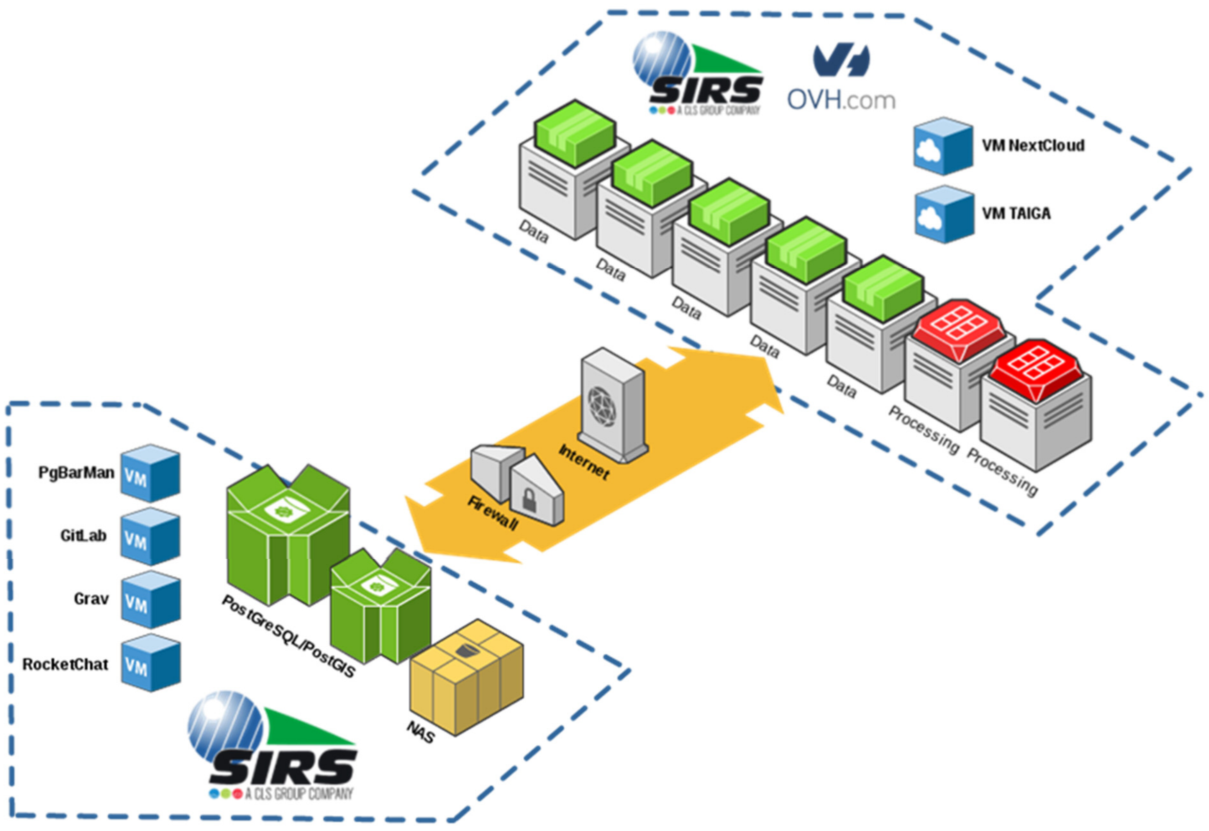


Figure 8: Overview of Service Infrastructure and Architecture Setup at SIRS

A second processing platform utilizing hardware and services of OVH (a French cloud computing company) offers extended storage and processing capabilities.

In general SIRS offers processing tools for raster and vector data. Both contain automated processing modules as well as applications for workflows with user interaction

Raster tools are based on several Open Source applications and libraries:

- QGIS processing linked to GRASS, GDAL and OTB
- SNAP (ESA software)
- GDAL with Linux Bash automation

The vector tools are based on popular Open Source applications and the corresponding interfaces:

- QGIS/GRASS algorithms
- PostGIS functions

2.5.3 EO data processing - Experiences from DLR

The Earth observation centre (EOC) at DLR has two large-scale processing environments, the Geofarm and Calvalus system.

The Geofarm in its current configuration consists of 2 Blade centres hosting 16 servers with a total of 672 Opteron cores, 3.3TB RAM, 488TB SATA and 50TB SAS storage interconnected with 10Gb/s Ethernet. In addition to that the setup also includes human resources for coordination, integration, configuration and virtualization of the environment. Within the next two years the system will encompass 4300 cores, 33TB RAM, 1.9PB SATA, 100TB SAS and 8TB SSD. The standard service footprint allocated for users at the Geofarm includes 2 virtual machines with 4 cores, but can be expanded to the full capacity on-demand.

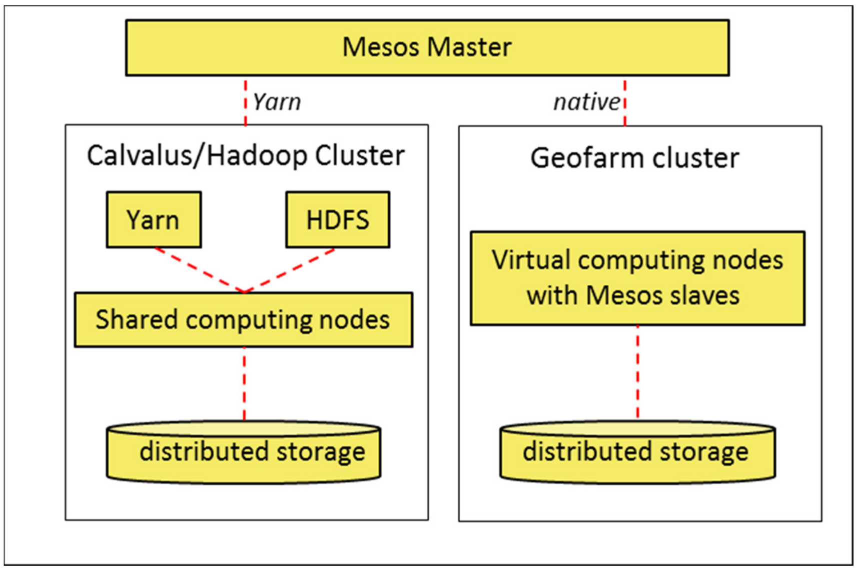


Figure 9: Overview of the two processing centres in the EOC at DLR

The DLR Calvalus Cluster was acquired in the frame of the OPUS project and delivered by Brockman Consult. It is a combined hard- and software stack based on Hadoop MapReduce. It consists of 50 compute nodes with 32 GB RAM each and 1 quad-core Intel Xeon 3.4 GHz CPU, and a distributed file system with 1PB. The Calvalus system extends and adapts the classical MapReduce paradigm with tools that are geared for EO data processing. The nodes are controlled by master running Hadoop Yarn. Data ingestion is managed by single point of entry called the feeder node that distributes data into the HDFS file system of the cluster. Calvalus integrates the SNAP and Sentinel toolboxes and allows developing own modules using Java, Python and Bash. A major capability of Calvalus is the integration of Docker containers. There are a set of tools for deploying and managing Docker container, which in turn are orchestrated via the central YARN master. This flexibility is extremely important in a research environment where people use a large set of programming languages and libraries and like to test prototypes and run immature code. Ordinary dependency management would quickly smudge and clog the single processing nodes with unnecessary and conflicting tools and libraries.

In order to trigger request for processing users can use any Web processing service (WPS). The WPS OGC standard allows submission of requests as XML and REST calls. The WPS requests are then serialized and ingested by a Mesos master, an Open Source resource management and scheduling system. The Mesos master knows the state and controls all available computing resources; in case of DLR the Geofarm and Calvalus clusters. The master checks availability of suitable processing resources (Mesos slaves) and triggers the calculations. In case of the Geofarm cluster each virtual machine is managed as one Mesos slave, whereas the Calvalus cluster is treated as one giant resource that is scheduled via the Mesos-YARN interface. Once the job is finished and the results are transferred to the distribution nodes, the master is informed by the corresponding slave that processing is finished and resources are ready for new tasks. This system ensures flexible and transparent management of a heterogeneous processing infrastructure. It also allows the flexible integration of additional processing nodes.

2.5.4 EO data processing at Joanneum

The IT infrastructure at Joanneum comprises a NAS system with a capacity of 50TB and a cluster of five Linux nodes for processing. In addition to that there is further disk space on a second workstation and the possibility to integrate local, idle desktop PCs for processing.

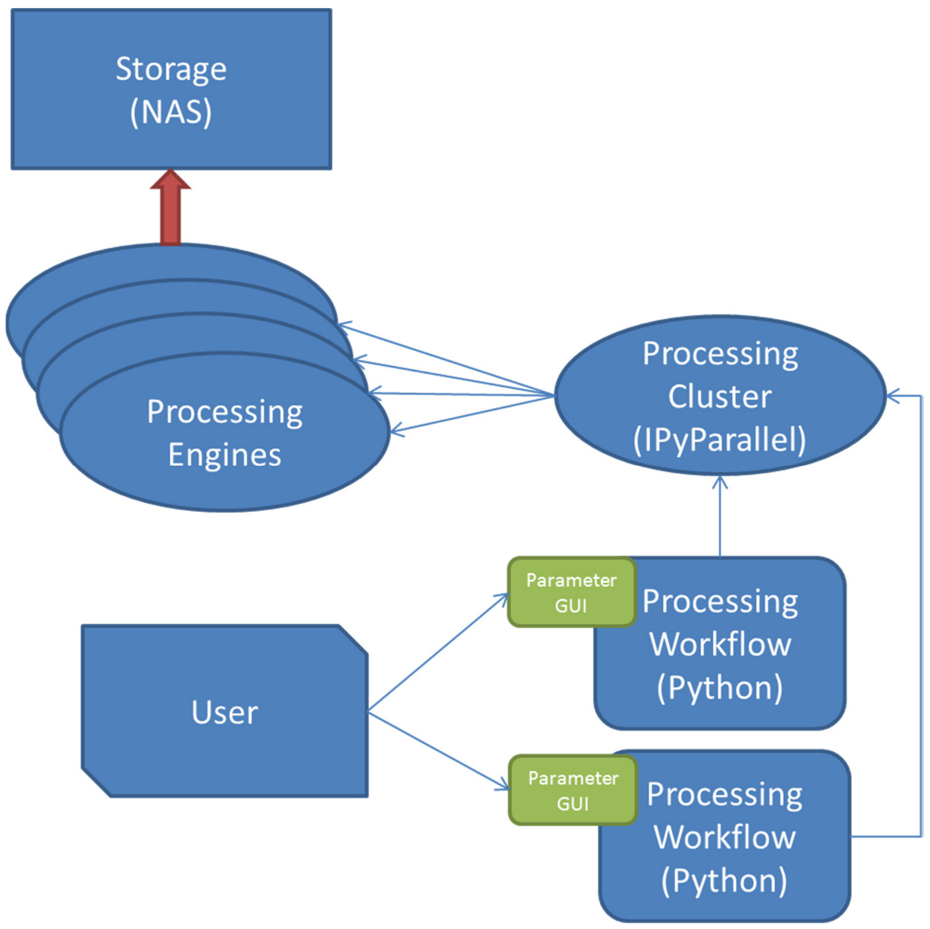

Storage (NAS)
Processing Engines
Processing Cluster (IPyParallel)
Parameter GUI
Processing Workflow (Python)
User
Parameter GUI
Processing Workflow (Python)

Figure 10: Overview of a processing work flow at Joanneum

Software development is tightly linked to two commercial processing suites, one for optical and one for radar data. The processing suites offer a wide range of algorithms and tools and can be combined in workflows, which are executed in parallel on the processing cluster.

3 Assessment of future developments within the project

3.1 Analysis of processing platforms

The WP 23 description called for an analysis of available and upcoming IT infrastructures in the frame of the Copernicus program, especially in regard to:

- network/data transfer capabilities
- storage architecture
- processing capacities, scalability and interfaces

In order to do that this document outlines a guide for a system evaluation that will be followed whenever a target platform is evaluated. The evaluation results will be matched with specifications of processing tools created to produce the data sets specified in ECoLaSS tasks 3 and 4. The purpose of the second analysis is supposed to influence the design of the processing tools in order to effectively run on the target platforms. **The result of system evaluations and analysis will be presented in deliverable D23.1b.**

3.1.1 Initial estimates

In order to obtain some ideas regarding data volumes during processing, some simple estimates might shed some light on basic processing constraints.

Data transfer is usually expressed in Mbits/s, meaning that a 1 Gigabit/s connection is able to transfer 125MB/s. Tests using Linux command line tools like the ones below

network only, 1Gbit line

```
      dd bs=1024 count=10000000 if=/dev/zero | ssh $targetIp dd of=/dev/null
      → 10240000000 bytes (10 GB, 9,5 GiB) copied, 98,8508 s, 104 MB/s
```

network and writing, 1Gbit line

```
      dd bs=1024 count=10000000 if=/dev/zero | ssh $targetIp dd of=outfile
      • 10240000000 bytes (10 GB, 9.5 GiB) copied, 123.495 s, 82.9 MB/s
```

to measure the real transfer volume between two given nodes show that the theoretical limit is almost completely utilized for one or few connections. In our example case it is safe to assume that the theoretical transfer rate is equivalent the real one.

During 2017 an average S2 granule in Central Europe contained 80 scenes with approximately 50 GB of data. That means that each granule grows by 1 scene of 0.5GB every 3 days, clearly indicating that processing single scenes poses no problems. The complexity grows whenever processing deals with time series or larger regions, since a granule covers only 100x100km.

Now let’s consider processing a time series of one year for a single S2 granule: Transferring a single S2 granule for a given year worth of 50GB over a 1Gbit line takes about 7 minutes, excluding reading and writing, which strongly depend on the corresponding storage medium. Atmospheric correction averages 55 minutes [1] per scene for reading, calculating and writing, or 3 days for the entire granule. During that operation, the data volume grows by 40 percent to 70GB, which require another 9 minutes of transfer to the output destination. It seems that data transfer is a minor issue (15 minutes transfer vs. 3 days of processing), especially since this case only considered pre-processing and neglected all the subsequent processing steps necessary to create any product.

However, this deliberation has a major flaw: It assumes that band-width and storage are uncontested whereas processing is. That changes quickly if multiple parallel processes simultaneously try to access storage via a network connection with a fixed maximum value regarding data volume or number of file handles. This can virtually bring processing to a standstill. Therefore, it is of vital importance that processing systems are designed in way that parallel, concurrent processing does not lead to network starvation.

Any processing system suitable for EO calculations on a continental or global scale must be able to offer at least several dozens to hundreds computing nodes that can read and write concurrently from/to a storage area without overwhelming the network or associated storage devices. This can be achieved with many different hardware architectures, as was already outlined in chapter 2.

3.1.2 Evaluation of a processing system

The purpose of this chapter is to give a detailed understanding and definition of the system evaluation for a processing framework.

In order to have a common understanding the following terms will be used accordingly:

[1] This is the average processing time of approximately 600 scenes from several Central European granules using sen2cor (latest version 2.4.0) on a Hadoop cluster, where transfer is not relevant. Processing times range from a few minutes to almost 3 hours.

Processor:	A Processor is an application startable in the course of a data processing Workflow. This could be an operational implementation of a scientific algorithm.
Request:	A Request is the unit of processing of data. A request is a set of Steps which form a complete workflow which creates output products from input Products.
Step:	A Step is the smallest unit of a processing request. It can be e.g. the startup of a Processor as well as the allocation of cache space or the transfer of a product between systems.
Workflow:	A Workflow describes the sequence of events needed to fulfil a Request. It can be depicted in the form of a tree diagram which includes Steps, decision-Steps and possible interconnections between these.
Resource:	A Resource is an allocateable object needed to fulfil the requirements of a Request.
Schedule:	A Schedule is a plan that includes a sequence of Steps and allocated Resources for these Steps.
Processing Power:	Processing Power is the number of Threads supported by a Resource/required by a Step.
Operator:	An Operator is a user of the processing system which handles off-nominal situations as well as unavailability of Resources etc. but also has control of the nominal task fulfilment.
Ingestion:	Ingestion is the process of registering and archiving a Product.
Administrator:	An administrator is an actor responsible for the hardware and software platform.
Integrator:	An Integrator is a user of the processing system framework implementing Workflows.
Instance:	An Instance is a concrete incarnation of an abstract definition.
Module/Modular:	A Module is a separately runnable element of the System. Modular design facilitates exchange of module-implementations by defining appropriate interfaces.
Product:	A Product is the result of a Step.

It is assumed that a processing centre consists of some of the hard and software resources shown in the image below:

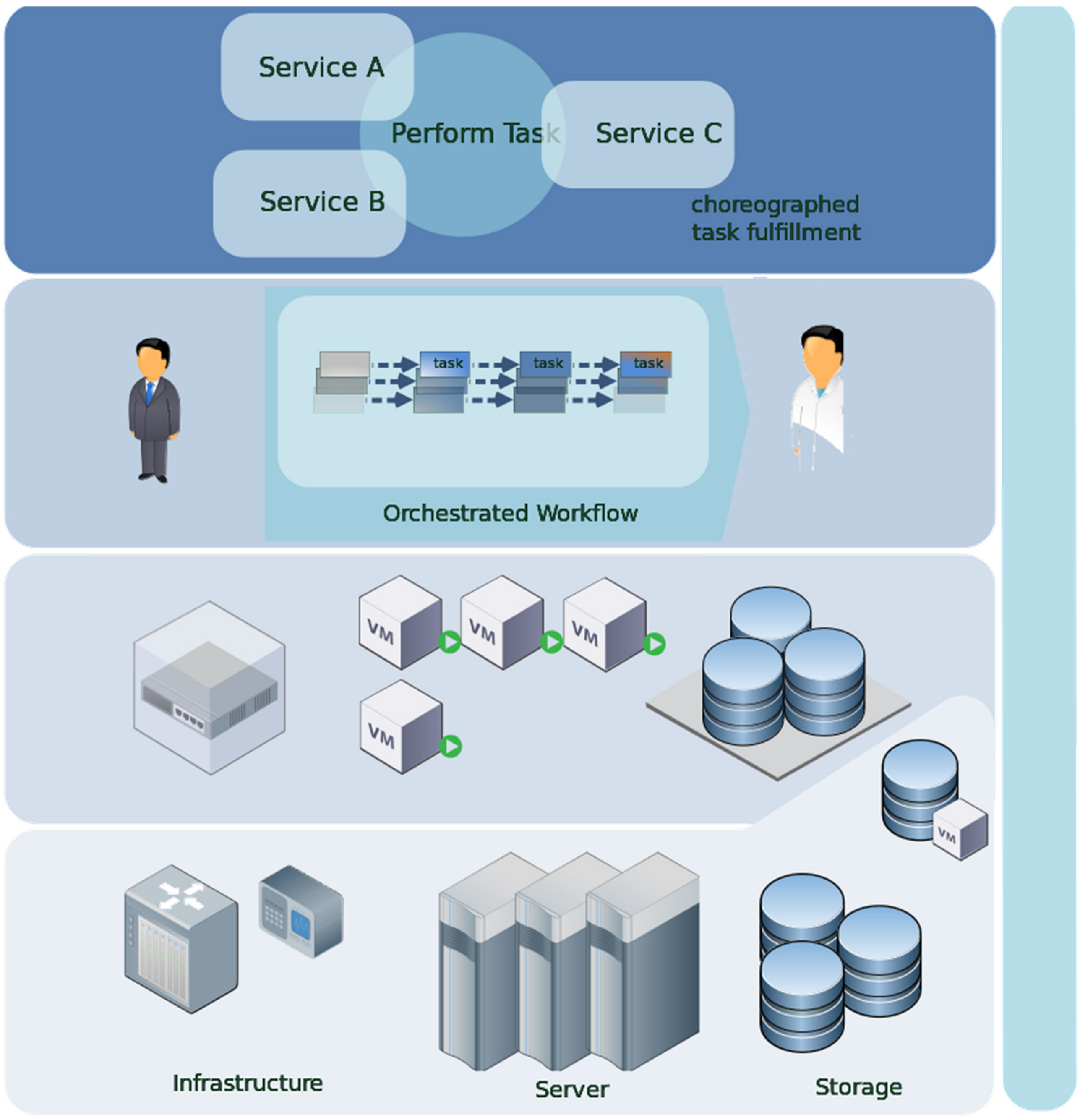


Figure 11: Dynamic approach for hardware and software resource allocation

Today most processing environments are based on a virtualized infrastructure. This facilitates a disconnection of hardware acquisition and hardware allocation for any project. The concept of separated resources can be transferred to the application layer as well. In figure XX the introduction of an abstraction layer between hardware and application – a dynamic infrastructure – is shown. With respect to the application layer an introduction of a dynamic layer separates project specific workflows from non project specific resource allocation tasks.

Processing management in this context has an essential role. It needs to provide a degree of freedom to facilitate inter-project resource usage as well as the means to easily create new processing workflow implementations.

Typical use cases of a processing environment are:

a) Ingestion of single products
b) Full automatic systematic data driven processing
c) Request driven processing
d) Iterative workflow handling
e) Mass-data processing
f) Validating parallel processing
g) Ad-hoc processing
h) Processing with operator interaction
i) Near real time processing

In our case we do not need to be concerned with point a) since we assume that all the required data all already present and accessible.

Typical, closely related use cases are b), c) and d), which describe processing tasks beginning with the availability of data to a defined end product (b), a specific request triggered to obtain a defined end product (c) and the repeated processing of a data set until a certain degree of maturity is achieved (d).

Topic e), request driven processing of a large number of products assumes that data stored in an archive and need to be transferred in smaller parts to the processing cache. After processing the output could be transferred to the archiving system or to other places.

A problem is the possible overload of the archiving system (read/write), network overload and the cache- and resource- management of the processing system. The cache and the processing resources are normally not powerful enough to handle all productions at the same time. These cases should be managed by a single mass processing request, which handles the whole processing request of the entire data set. In contrast to request driven processing, mass processing needs to manage all elements of the workflow depending on the available resources.

Cases involving parallel requests to validate and compare a new version with an older one (f), ad-hoc requests of processors for test purposes with only limited configuration (g) and automatic processing until a step requires interaction of the operator (h) will not be considered in detail since these cases are only important for the management of the system.

Finally, in case i) - near real time processing (NRT) - the processing time window is the most imported condition. The problem is the mix of NRT and non NRT tasks in processing facility, which shares the resources of the system. Here the processing facility must be able to prioritize the usage of the resource. In critical NRT situations the processing facility must even be able "steal" the resources from low priority processes.

3.1.3 Checklist for evaluation

With these use cases in mind the processing environments will be evaluated according to the following criteria falling into several groups:

Design Constraints

This assumes that the processing environment is a modular architecture including interfaces to internal and external systems.

- Are interfaces present
- Are interfaces open
- Are there interfaces for processor integration
- Are there interfaces that provide a complete set of information about a processing run

- Are there interfaces for workflow construction
- Are there interfaces for setting breakpoints in workflows
- Are there interfaces for local, distributed and remote storage
- Are there interfaces for version controlled
- Are there restrictions/quotas/limits regarding processors and resources

Resource Handling

A core functionality of a processing framework is the handling of resources like disc space, CPUs and memory.

- Is storage space handled as a resource
- Can a resource like storage be associated to workflow or single workflow steps
- Can memory be allocated before hand
- Are products handled as a resource
- Are processors handled as a resource
- Can processors be transferred to data
- How are product resources assigned processor resources
- How are CPUs allocated to processes
- How is memory allocated to processes

Scheduling

To be able to handle more than one request a time a processing environment provides automatic scheduling.

- Can product ingestion be scheduled
- What type of use cases (see above) are scheduled
- Can workflows be scheduled
- Can jobs be scheduled in parallel
- Can workflow steps be parallelized
- How are queues handled
- How are tasks selected and prioritized

Processing constraints

- Which of functions are supported by the processor interface:
  start/stop/status/restart/suspend/resume
- Can a processor trigger another request
- Does the processor interface return status reports, log messages or return values
- Is there a mechanism for debugging
- How are failures treated by the system
- Are processing task logged and archived
- Are there quotas for memory, CPU, storage
- Are there user right management constraints

3.2 Preliminary recommendations

As mentioned above the main conclusions will be presented in deliverable WP23.1b where the evaluation scheme above will be matched against potential processing environments. The result of each evaluation will be used to formulate recommendations regarding the design of any processors developed in this project. But it is possible to come up with some preliminary recommendations.

The previous chapters show that processing infrastructures within the consortium are extremely heterogeneous. They reflect the historical development and current emphasis of each entity.

SIRS and DLR have access to large connected hardware clusters that allows them to store and find data across multiple nodes as well as process data on some or all of the nodes using dedicated process management systems. Joanneum is strongly focused on software development and maintains several smaller and isolated processing clusters, which are orchestrated using a series of scripts developed in house. They can afford this approach because they have a suite of software tools, that operates in a specific manner and thus need less flexibility, whereas software tools present at e.g. DLR are from multiple development teams. Thus, they are very heterogeneous and must be used in a different manner requiring a processing management tool with a large degree of flexibility.

In the course of the project the system design approach outline before will need to identify the key resources of each processing platform and evaluate them in respect to a general set of requirements derived from potential target platform like DIAS.

As mentioned above IT infrastructures of the consortium members are diverse, resulting from different needs and requirements. The same is valid for the software used for analysing data and managing infrastructure. But the ECoLaSS project could benefit from a few design considerations and maybe some activities related to collaboration in the field of prototype development and processing exercises. Here are some considerations:

HARDWARE INFRASTRUCTURE

The advent of virtualization techniques and cluster management tools allows users to avoid buying powerful, but expensive hardware geared for high level processing and obtain ordinary, run of the mill computers. New techniques allow combining single resources into larger processing entities. At the same time, one has to be aware that certain processing tasks might require the procurement of special hardware (e.g. machine learning favours GPU processing units). However specialized systems might be replaced by utilizing services from third parties like commercial service providers and public computing centres.

It is even possible to utilize single, ordinary desktop computers connected via intranet. Projects like the Berkeley Open Infrastructure for Network Computing (BOINC) try to employ unused and idle computers for scientific processing. Therefore, a client on each computer registers the activity and notifies a master if it is ready to perform additional jobs. The master then assigns data and software to the node and thus distributes processing tasks. Distributed computing projects like that are similar to some of the systems described above. Unfortunately, they are quite rigid regarding the choice of software and data structures.

Since hardware setup at each consortium member is fixed no recommendations can be given for any collaboration activities.

SOFTWARE INFRASTRUCTURE

Software infrastructure is probably an area where differences across and maybe even within institutions are extremely large. The choice of programming languages and tools, the level of operationalization, skill and number of programmers and many more influence the functionality and capability of software on numerous levels.

It is therefore really difficult to give recommendations for collaboration. Some ideas might include:

- Definition of common product specifications
- Usage of a set of common libraries

- Sharing of code for common tasks (import, export, transformation, algorithms)
- Definition of common APIs
- Provision of web services for data (WMS, WFS, and other OGC services) and functions
- Exchange of virtual images or containers (see below)

PLATFORM INFRASTRUCTURE

In contrast to the subject covered above platform infrastructure holds big promises for exchange of software. Central to this is the last topic in the list mentioned above: Virtualization techniques enable stakeholders to exchange data, software and services without dependencies regarding applications, libraries and languages except for the virtualization system itself. Thus, they offer an isolation of the underlying hardware or operating system and permit encapsulation of entire applications in a virtual or container image.

Establishing processing platforms that can handle virtual images or containers allows platform providers to invite users to work with their preferred collection of tools without the necessity to change the setup of the host system. Images can even be replicated multiple times for parallel processing since the footprint usually is negligible. It is thus possible to distribute software to one (or multiple) processing centre, where it can efficiently analyse data sets present on the host platform. Testing is also simplified because new software can be incorporated in a container and subsequently deployed on the host system. Different versions of the software can be tested next to each other without interference or changes of the underlying system.

A less flexible way is the usage of platforms where the user has to adhere to the dependencies of the host system. This might be an API for services, a set of languages and libraries, or even a series of applications. In this case the user has to use the technique provided by the host provider, but is at least able to write his own software. During the last years this is becoming easier, because of the development of some de facto standard libraries (e.g. GDAL for I/O operations) and languages like Python or R that are used by many programmers in the field of data sciences.

A combination of containers and standards and conventions regarding their execution (e.g. API regarding “docker run” calls) provides maximum flexibility. In this case different parties can contribute data manipulation tools to platform service providers who can rely on the conventions to automatically deploy and run them on the host platform.

Exchanging images for certain processing tasks (e.g. pre-processing) is considered to be one of the least intrusive and therefore most promising forms of collaboration within the consortium.

OVERARCHING ORCHESTRATION

In simple cases orchestration can be achieved by the command line. But once processing becomes more complex users will quickly look for ways to automate monitoring resources, conditionally manage job requests (e.g. construct workflows) and check the status of completed processing tasks. An even more complex scenario involves coordination of one or more remote processing services. In order to be able to orchestrate a group of tasks it becomes imperative to define an API or at least provide conventions for calling these processing tasks.

It has become very popular to furnish single tasks with web services based on protocols like REST or WPS. Orchestration tools usually allow the construction of adapters that interact with established services and some popular Open source tools already offer many built-in connectors for the most common protocols.

It would be interesting to investigate if orchestration across services provided by some project members could speed up certain processing tasks, especially if one could render transfer of large data sets obsolete.

4 Outlook and recommendations

As mentioned throughout this report, developments in the IT domain during the last 10 years have significantly shifted our perception of the way we use large data sets. Data volumes produced by companies like Google, Twitter, Amazon, Facebook and others are huge and not only constitute business core data, but also protocols and logs of people using these services. It should be noted that the (daily) volumes of EO data generated by space agencies and commercial providers are minor compared to real-time data of large IT corporations. These companies not only want to store but also evaluate and analyse real-time data. The resulting processing concepts and accompanying collaborative, Open Source implementations also benefit other fields in the IT community.

It is also important to realize that the expertise and infrastructure of IT corporations is centred on hardware and software techniques related to large scale computing. This is usually not the case with classical EO data users comprising universities, research institutes, government agencies and small to medium sized companies. They typically developed their IT knowledge from bottom up to match the tasks at hand. For a long time, these users did not have a global perspective. But it is imperative to look at the latest developments and try to benefit from them.

In particular users will have to decide when to process data locally or remotely using services in the clouds. In order to stay flexible, processing techniques will favour solutions that move software to data. This opens the way for collaborations with partners and keeps stakeholders independent of hardware and service providers. Other considerations (security, confidentiality, technical requirements, etc.) might lead to in-house solutions, but new technical concepts already offer many building blocks to setup a maintainable, inexpensive and flexible processing environment.

Preliminary experience of consortium members and chapter 3.1.1 show that the EO community does not necessarily need a high performance infrastructure but something known as "large scale analytical platform" that enables users to process multiple data sets in parallel without running into IO squeezes.

5 References

Amazon Web Services, Inc. (2017). Retrieved 12 18, 2017, from Earth on AWS: https://aws.amazon.com/earth/

Apache. (2013, 04 08). Retrieved 12 18, 2017, from HDFS Architecture Guide: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

Apache. (2017). Retrieved 12 18, 2017, from Apache Mesos: http://mesos.apache.org/

Apache Inc. (n.d.). Retrieved 12 18, 2017, from Project - Airflow Documentation: https://airflow.apache.org/

Canonical Ltd. (n.d.). *LXC Introduction*. Retrieved 12 18, 2017, from Linux Containers: https://linuxcontainers.org/lxc/introduction/

CNES - Centre National d'Études Spatiales. (2017). Consulté le 12 18, 2017, sur PEPS - Plateforme d'Exploitation des Produits Sentinel (CNES): https://peps.cnes.fr/rocket/#/home

Docker Inc. (2017). Retrieved 12 18, 2017, from Docker - Build, Ship, and Run Any App, Anywhere: https://www.docker.com/

EODC Earth Observation Data Centre for Water Resources Monitoring GmbH. (2017). Retrieved 12 18, 2017, from We bring the Software to the Data and Information Products to the Users: https://www.eodc.eu

ESA. (2017). *Existing/Planned collaborative ground segment.* Retrieved 12 18, 2017, from Sentinel Online: https://sentinel.esa.int/web/sentinel/missions/collaborative/existing-planned

ESA. (2017, a). Retrieved 12 18, 2017, from TEP - Thematic Exploitation Platform: https://tep.eo.esa.int/

ESA. (2017, b). Retrieved 12 18, 2017, from Research and User Support: https://rus-copernicus.eu/portal/

European Commision. (2017, 12 14). *Copernicus DIAS contracts signed*. Retrieved from Copernicus: http://copernicus.eu/news/copernicus-dias-contracts-signed

European Commission. (2017). *The upcoming Copernicus Data and Information Access Services (DIAS).* Retrieved 12 18, 2017, from Copernicus: http://copernicus.eu/news/upcoming-copernicus-data-and-information-access-services-dias

Ghemawat, S., Gobioff, H., & S.T., L. (2003). The Google File System. *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles - SOSP '03*.

Google. (2017). Retrieved 12 18, 2017, from Google Earth Engine: https://earthengine.google.com/

IBM. (n.d.). *IBM Spectrum Scale - Overview*. Retrieved 12 18, 2017, from https://www.ibm.com/us-en/marketplace/scale-out-file-and-object-storage

Infrastruktur, BVDI - Bundesministerium für Verkehr und digitale. (2017). Retrieved 12 18, 2017, from CODE-DE: https://code-de.org/

Jenkins Project. (n.d.). Retrieved 12 18, 2017, from Jenkins: https://jenkins-ci.org/

Linux Foundation. (2017). Retrieved 12 18, 2017, from Kubernetes | Production-Grade Container Orchestration: https://kubernetes.io/

NOA - National Observatory of Athens. (2016). Retrieved 12 18, 2017, from Hellenic National Sentinel Data Mirror Site - Overview: https://sentinels.space.noa.gr/

Oracle. (2012). *What is ZFS?* Retrieved 12 18, 2017, from https://docs.oracle.com/cd/E23823_01/html/819-5461/zfsover-2.html#gayou

Red Hat, Inc. (2017). Retrieved 12 18, 2017, from Ansible Is Simple IT Automation: https://www.ansible.com/

SAP. (2017, 12 18). *Intro to SAP HANA Geospatial - Spatial Reference Systems | SAP*. Retrieved 12 18, 2017, from https://www.sap.com/developer/tutorials/hana-spatial-intro6-srs.html

SERCO Spa. (2016). *Sentinels Data Access Annual Report 2016.* ESA. Retrieved 12 18, 2017, from https://scihub.copernicus.eu/twiki/pub/SciHubWebPortal/AnnualReport2016/COPE-SERCO-RP-17-0071_-_Sentinel_Data_Access_Annual_Report_2016_v1.2.pdf

Thulasiraman, K., & Swamy, M. (1992). 5.7 Acyclic Directed Graphs. In *Graphs: Theory and Algorithms* (p. 118). John Wiley and Sons, Inc.

VITO - Vlaamse Instelling voor Technologisch Onderzoek. (2017). Retrieved 12 18, 2017, from Terrascope | EO needs in Belgium: https://www.terrascope.be/