THE EXTREMEEARTH SOFTWARE ARCHITECTURE FOR COPERNICUS EARTH OBSERVATION DATA

Desta Haileselassie Hagos[1], *Theofilos Kakantousis*[2], *Vladimir Vlassov*[1], *Sina Sheikholeslami*[1]
Tianze Wang[1], *Jim Dowling*[1, 2], *Andrew Fleming*[3], *Andreas Cziferszky*[3], *Markus Muerth*[4], *Florian Appel*[4]
Despina-Athanasia Pantazi[5], *Dimitris Bilidas*[5], *George Papadakis*[5], *George Mandilaras*[5]
George Stamoulis[5], *Manolis Koubarakis*[5], *Antonis Troumpoukis*[6], *Stasinos Konstantopoulos*[6]

[1] KTH Royal Institute of Technology [2] Logical Clocks AB [3] British Antarctic Survey [4] VISTA GmbH
[5] National and Kapodistrian University of Athens [6] National Center for Scientific Research - Demokritos

ABSTRACT

Current deep learning architectures for remote sensing are trained on small datasets typically using 1 GPU without taking advantage of new innovative approaches such as distributed scale-out deep learning. In this paper, we present the ExtremeEarth software architecture for Copernicus Earth Observation data. We show how we go beyond the state-of-the-art by scaling to the petabytes of data using Hopsworks and demonstrate our big data technologies in two Thematic Exploitation Platforms (TEPs): Food Security and Polar. Furthermore, we present the integration of Hopsworks with the Polar and Food Security use cases and the flow of events for the products offered through the TEPs.

***Index Terms*—** Copernicus, ExtremeEarth, Hopsworks, Earth Observation, Linked Geospatial Data, Deep Learning

1. INTRODUCTION

The expanding operational capability of global monitoring from space, combined with data from long-term Earth Observation (EO) archives provides users with unprecedented insights into how oceans, atmosphere, land, and ice operate and interact as part of an interconnected earth system. The availability of the growing volume of environmental data from space represents a unique opportunity for science and applications. Nevertheless, the task of achieving its full potential in terms of data exploration is a major challenge associated with this opportunity. In this paper, we present the ExtremeEarth software infrastructure that builds on seamless integration of both existing and novel software platforms and tools for storing, accessing, processing, analyzing, and visualizing large amounts of Copernicus EO data. The knowledge extracted from the satellite data is then encoded as linked geospatial data and integrated with other open linked data sources. Relevant technologies enable both EO and non-EO expert users to pose queries over this information in order to develop prototype environmental and business applications. One of the main objectives of the ExtremeEarth project is the development of distributed Deep Learning (DL) classification architectures tailored to the specific properties of Copernicus EO data. The information provided by the Copernicus satellite data is then used for knowledge extraction, focusing on the following two use cases.

Polar Use Case. This use case exploits long time series of Sentinel-1 Synthetic-aperture radar (SAR) images to develop processing architectures and algorithms to cope with the extreme analytics and big data challenges associated with sea ice monitoring [4].

Food Security Use Case. It aims at the assessment of high-resolution water availability for fertilisation and irrigation by combining long time series of Sentinel-2 multispectral images with crop growth modeling to provide water availability. The DL architecture presented for this use case enables a processing chain that generates crop type and crop boundaries maps, using Sentinel-2 data for training.

The rest of the paper is organized as follows. Section 2 presents the components of ExtremeEarth platform software architecture, while their integration is presented in Section 3. The ExtremeEarth platform's flow of events is demonstrated in detail in Section 4. Finally, Section 5 concludes our paper.

2. SOFTWARE ARCHITECTURE

This section presents a detailed overview of the high-level components of the ExtremeEarth platform architecture. As shown in Figure 1, the ExtremeEarth infrastructure consists of *four* layers with the TEPs in the *product layer*, Hopsworks [5] in the *processing layer*, CREODIAS[1] in the *data layer*, and Infrastructure as a Service (IaaS) in the *physical layer*.

Product layer. The product layer provides a collaborative virtual work environment, through TEPs, that operates in the cloud and enables access to the products, tools, and relevant EO, and non-EO data.

This work is supported by the ExtremeEarth project funded by European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 825258.

[1] https://creodias.eu/

Processing layer. This layer provides the Hopsworks data-intensive Artificial Intelligence (AI) platform. Hopsworks is an open source platform that provides an execution environment for data science and data engineering, which supports the design, distributed training and operation of end-to-end Machine Learning (ML) pipelines.

Data layer. The data layer offers a cloud-based platform that facilitates and standardizes access to EO data through a Data and Information Access Service (DIAS). It also provides centralized access to Copernicus data and information, as well as to processing tools. TEPs are installed and run on a DIAS infrastructure, which in the case of ExtremeEarth is the CREODIAS.

Physical layer. It contains the cloud environment's compute, storage, networking resources, and hardware infrastructures.

3. INTEGRATION OF COMPONENTS

In this section, we define the semantics of the integration of Hopsworks with the Polar and Food Security TEPs in ExtremeEarth. The APIs used for the full integration of the ExtremeEarth components via the inter-layer interfaces of the software platform are also described in this section. Figure 1 contains the numbered labels highlighting the different integration points of the DIAS, TEPs, and Hopsworks. Integration includes defining the user roles and processors that have to be implemented to provide a seamless experience for ExtremeEarth users. This integration is split into two iterations; the first iteration develops functional products that will satisfy the requirements of the use cases, while the second iteration builds on the first one to automate as many of the manual processes as possible. A detailed description of each integration point is provided in the following.

1. **Raw EO data**. DIASes and CREODIAS in particular, provide EO data access to services and applications running on their platforms. This includes the downstream of Copernicus data as it is generated by satellites, such as the Sentinel constellations. At an infrastructure level, this data is persisted at an object store with an *S3* object interface, managed by CREODIAS. This infrastructure comes with *s3fs*, a tool that emulates filesystem storage using an S3 object interface [1].

2. **Pre-processed data**. TEPs provide the developers of the DL pipelines with pre-processed EO data which forms the basis for creating training and testing datasets. This pre-processed data is managed by TEPs but is stored in the CREODIAS infrastructure. Developers of the DL pipeline can also define and execute their own pre-processing, if the pre-processed data is not already available. To do that, various third-party libraries and frameworks can be implemented and used in different programming languages such as Python, C++ and Java. To further assist ExtremeEarth users with pre-processing data, Hopsworks provides APIs to run programs in languages other than Python, which is the de-facto language of ML frameworks and libraries.

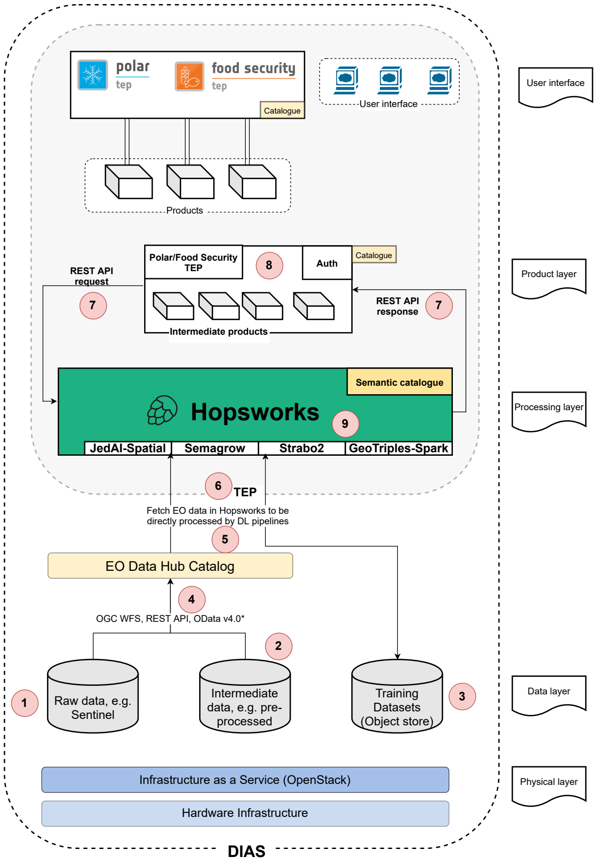

Fig. 1: Overview of ExtremeEarth platform infrastructure.

3. **Object store**. CREODIAS provides an object store used for storing data produced and consumed by the TEPs services and applications. In ExtremeEarth, this object store is used primarily for storing training data required by the Polar and Food Security use cases. This training data is provided as input to the DL pipelines.

4. **EO Data Hub Catalog**. This service is provided and managed by CREODIAS. It provides various protocols including OGC WFS, a REST API, and OData [2] as interfaces to the EO data. The Data Hub uses cloud infrastructure and methods to efficiently process and distribute data in a matter of seconds. It removes the major hassle of downloading, archiving, and processing petabytes of data and simply makes the full and global archive easily available immediately via web services. EO Data Hub technology is designed to work with original EO data, avoiding the need for computing-intensive pre-processing and additional storage for processed tiles [2].

5. **TEP-Hopsworks EO data access**. Users can directly access raw EO data from their applications running on Hopsworks. Multiple methods, e.g., object data access API (*SWIFT/S3*), filesystem interface, etc., are provided for accessing Copernicus and other EO-data available on CREODIAS.
6. **TEP-Hopsworks infrastructure integration**. Hopsworks and both the Polar and Food Security TEPs are installed and operated on a DIAS infrastructure, which in the case of ExtremeEarth is the CREODIAS infrastructure. CREODIAS and its administrative tools enable TEPs to spawn and manage virtual machines and storage by using CloudFerro which provides an OpenStack-based cloud platform to TEPs. Hopsworks is then installed on virtual machines within the TEP domain so that it can access compute and storage resources provided by the TEPs.
7. **TEP-Hopsworks API integration**. Hopsworks is provided within the TEP environment as a separate application and is mainly used as a development platform for the DL pipelines and applications of the Polar and Food Security use cases. These applications are exposed in the form of processors to the TEP users. Practically, a processor is a TEP abstraction that uses ML models that have previously been developed by the data scientists of the ExtremeEarth use cases. Hence, TEP users are presented with a geographical map where they can select data from a specific region and timeframe as input to one of the available processors. This means that the ML model is developed within Hopsworks and then served to users as a processor in the TEP where they can make use of increased data access and options for batch or triggered processing capabilities.
8. **Hopsworks-TEPs datasets**. TEPs provide users with access to data to be served as input to processors from various sources. Such sources include the data provided by CREODIAS and external services that the TEP can connect to. For the Polar use case, Sentinel-1 data is the primary data source that is readily available on CREODIAS. For the Food Security use case, pre-processed Sentinel-2 data is the primary data source. The pre-processing takes place on the Food Security TEP, where cloud-masking, atmospheric correction, and extraction of leaf area index is performed before handing the data to Hopsworks. The pre-processed data is stored in an object storage provided by CREODIAS, thus made available to Hopsworks users by exchanging authentication information.
9. **Linked Data tools integration**. Linked data applications are deployed as Hopsworks jobs using Apache Spark. A data scientist, as shown in Figure 3, can use GeoTriples-Spark to transform big geospatial data into the Resource Description Framework (RDF) [9], according to the ontologies developed for the Food Security and the Polar use cases [3]. Then, JedAI-Spatial can be used for spatial interlinking. The resulting RDF data will be stored in Strabo2 [6], where the data scientist can pose GeoSPARQL queries using the Apache Livy interface for interacting with Spark. Moreover, a user can pose GeoSPARQL queries in Semagrow[2] to combine the big linked geospatial data served by Strabo2 with external geospatial sources. As an example, such integration allows the validation of Food Security data using external geospatial datasets [7]. Strabo2 and Semagrow will be deployed using the JBoss application server running in Hopsworks, and can be accessible as standard SPARQL endpoints from other modules of the Hopsworks platform, based on the SPARQL HTTP protocol [8].

Building and serving models in ExtremeEarth. The Polar and Food Security use cases are the two main scenarios of the ExtremeEarth software architecture for the application of large-scale Copernicus EO data. For the Polar use case, the Polar TEP has established an instance of the Hopsworks platform on CREODIAS to allow the development of DL models, as explained above. The resulting processor is integrated into the Polar TEP, as described above. For the Food Security use case, the instance of the Hopsworks platform on CREODIAS, set up for the Polar Use case, is jointly used. There are *two* methods by which the trained model can be served via the Polar TEP: *(i)* The model can be exported from Hopsworks and make it fully embedded into the Polar TEP processor. *(ii)* The model can be served online on Hopsworks and a processor on Polar TEP submits inference requests to the model serving instance on Hopsworks and returns the results. In method *(i)*, once the ML model is developed, it can then be transferred from Hopsworks to the Polar TEP by using the Hopsworks REST API and Python SDK. TEP users can integrate the Hopsworks Python SDK into the processor workflow to further automate the ML pipeline lifecyle. In method *(ii)*, Polar TEP is able to submit inference requests to the model being served by the online model serving infrastructure run on Kubernetes and hosted on Hopsworks.

Hopsworks provides a REST API for clients to work with model serving, and authentication is done in the form of API keys managed by Hopsworks on a per user basis. These keys can therefore be used by external clients to authenticate against the Hopsworks REST API. The TEP end-user would experience almost no difference between these two methods of model serving. However, the implementation is considerably different. Figure 2 shows how the ML model on Hopsworks can be linked to the Polar TEP processor. It shows how the classification step takes a patch or a set of patches (a batch) and transmits the image patches over the network to the Hopsworks REST API endpoint before the

[2] Cf. https://github.com/semagrow

model is served by the Hopsworks. The Food Security TEP is currently enhanced to establish connections to Hopsworks, to allow training and deployment of DL models as Food Security TEP services.

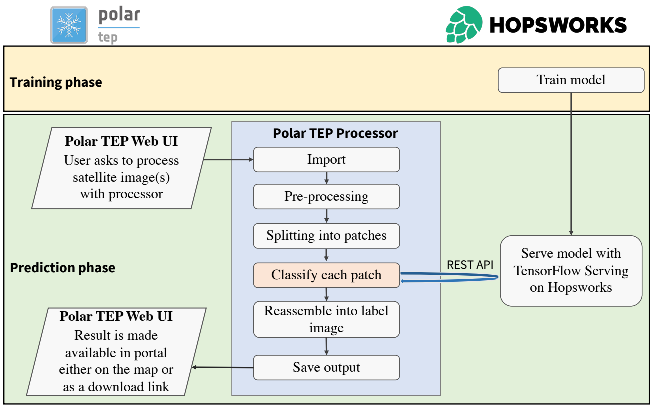
polar
tep
HOPSWORKS
Training phase
Train model
Polar TEP Web UI
User asks to process satellite image(s) with processor
Polar TEP Processor
Import
Pre-processing
Splitting into patches
Classify each patch
Reassemble into label image
Save output
REST API
Serve model with TensorFlow Serving on Hopsworks
Prediction phase
Polar TEP Web UI
Result is made available in portal either on the map or as a download link

Fig. 2: Interaction of the ML model on Hopsworks with the Polar TEP processor.

4. EXTREMEEARTH FLOW OF EVENTS

In this section, we demonstrate the ExtremeEarth platform architecture's flow of events. Figure 3 shows the flow of events of the products offered by the TEPs using Hopsworks. Details about the specific APIs and services used in each event is presented as follows:

(1) EO data scientists log in to Hopsworks, (2) Read and pre-process raw EO data in Hopsworks, TEP or in the local machine, (3) Create training datasets based on the intermediate pre-processed data, (4) Develop DL pipelines, (5) Perform linked data transformation, interlinking, and storage, (6) Log in to the Polar or Food Security TEP applications, (7) Select Hopsworks as the TEP processor. The processor starts the model serving in Hopsworks via the REST API. The processor also downloads the model from Hopsworks via the REST API and serving is done within the TEP application, (8) Submit federated queries with Semagrow and use the Hopsworks semantic catalogue.

5. CONCLUSION

ExtremeEarth software platform infrastructure builds on the seamless integration of existing and novel software platforms and tools for storing, accessing, processing, analyzing, and visualizing large amounts of Copernicus EO data. In this paper, we present the high-level components of the platform architecture and define the semantics of the integration of Hopsworks with the Polar and Food Security TEPs in ExtremeEarth. The APIs used for the full integration of the ExtremeEarth components via the inter-layer interfaces of the software platform and the platform architecture's flow of events are further demonstrated in this paper.

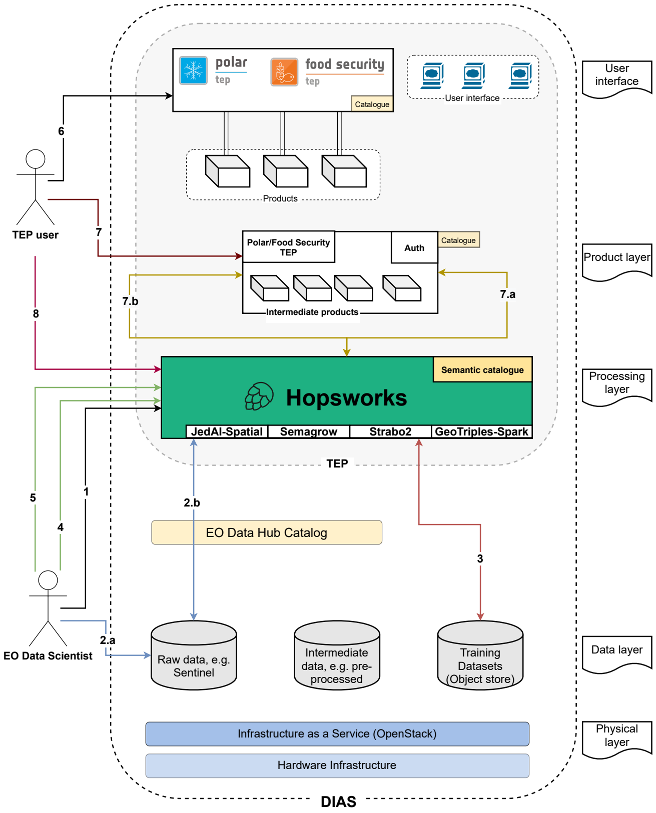

Fig. 3: ExtremeEarth architecture and flow of events.

REFERENCES

[1] S3FS. https://github.com/dask/s3fs/, 2021.

[2] CREODIAS. Data and Processing. https://creodias.eu/data-related-services, 2021.

[3] D.-A. Pantazi, G. Stamoulis, M. Koubarakis, N. Hughes, A. Everett, F. Appel. D1.7 - Semantic catalogue design and implementation-v2., 2020.

[4] C. O. Dumitru, V. Andrei, G. Schwarz, and M. Datcu. Machine Learning for Sea Ice Monitoring from Satellites. *Int. Arch. Photogrammetry, Remote Sensing & Spatial Inf. Sci.*, 2019.

[5] T. Kakantousis, A. Kouzoupis, F. Buso, G. Berthou, J. Dowling, and S. Haridi. Horizontally Scalable ML Pipelines with a Feature Store. In *Proc. 2nd SysML Conf., Palo Alto, USA*, 2019.

[6] M.Koubarakis et al. From Copernicus Big Data to Extreme Earth Analytics. In *22nd International Conference on Extending Database Technology, EDBT 2019; Lisbon; Portugal; March 2019*, pages 690–693. OpenProceedings, 2019.

[7] A. Troumpoukis, S. Konstantopoulos, and G. Mouchakis et al. GeoFedBench: A benchmark for federated GeoSPARQL query processors. In *Proc. Posters and Demos ISWC 2020*, 2020.

[8] W3C. SPARQL Protocol. https://www.w3.org/TR/sparql11-protocol/, 2013.

[9] W3C. Resource Description Framework (RDF). https://www.w3.org/RDF/, 2014.