MULTI-CLOUD PROCESSING WITH DASK: DEMONSTRATING THE CAPABILITIES OF DESTINE DATA LAKE (DEDL)

Christoph Reimer[1], Lukas Weidenholzer[1], Sean Hoyal[1], Bernhard Raml[2], Martin Schobben[2], Matthias Schramm[2], Wolfgang Wagner[1,2], Christian Briese[1]

[1] EODC Earth Observation Data Centre for Water Resources Monitoring GmbH

[2] Technische Universität Wien, Department of Geodesy and Geoinformation

Abstract

The DestinE Data Lake (DEDL) is one of the three key system components of the Destination Earth (DestinE) initiative of the European Commission. It is in the responsibility of the DEDL to provide seamless data access as well as support for big data processing services to DestinE users. One of the big data processing services offered via DEDL is based on the Dask ecosystem. The DEDL Dask service allows for multi-cloud processing, following the paradigm of data proximate computation. Digital Twins of the Earth will run on HPC system across Europe, producing petabytes of data per day over the next few years. Accordingly, enabling users to do processing close to the data is key to cope with these volumes of data in an efficient way. Those capabilities of the DEDL are presented by two use case demonstrations based on Pangeo core packages such as xarray and Dask.

Index Terms— Destination Earth, DestinE Data Lake, Big Data Processing, Pangeo, Dask, STAC, Flood, Drought

1. INTRODUCTION

The Destination Earth (DestinE) initiative of the European commission aims to create a Digital Twin of planet Earth. The initiative is implemented by ECMWF, ESA and EUMETSAT, each responsible for one key system component of DestinE. The DestinE Core Service Platform (DESP), entrusted to ESA, is acting as the main user facing component to interact with the DestinE system. ECMWF is responsible for the implementation of the Digital Twin Engine (DTE) to create digital replicas, Digital Twins, of the Earth system covering different thematic aspects. The DestinE Data Lake (DEDL) implemented by EUMETSAT is a central element in the system, acting as bridge between the DESP and the DTE [1]. The DEDL will provide a harmonised access to data from ESA, EUMETSAT, ECMWF, Copernicus, diverse federated data sources as well as outputs of the Digital Twins executed by the DTE. System requirements of the DEDL encompass the following key aspects:

- DEDL shall be built as a self-standing component, built from geographically distributed elements providing seamless access to required data to all DestinE users.
- Provisioning of data & information available from a large number of external data spaces or generated by the DestinE Digital Twins and applications on the Core Service Platform, regardless of data type and location.
- Support for near-data processing to maximize throughput and service scalability by utilising big data distributed workflows.

2. DESTINATION EARTH DATA LAKE

Requiring a self-standing component based on geographically distributed elements is enabled via the deployment of dedicated cloud infrastructures across Europe. The core infrastructure is deployed in Warsaw, Poland, referred to as Central Site, hosting all required services to be consumed by users via DESP and operator services to guarantee the operations of DEDL. The Central Site is complemented by so called Bridges, collocated to EuroHPC [14] environments powering the DTE to run Digital Twin simulations. At the current stage of the project, Bridges are established at Kajaani, Finland connecting LUMI supercomputer, and at Bologna, Italy close to the facilities of the LEONARDO supercomputer. LUMI and LEONARDO Bridge are installed to enable access to Digital Twin outputs stored in the Digital Twin (DT) Data Warehouse at the given EuroHPC locations.

The DEDL discovery and data access service provides a harmonised data access (HDA) to DTE outputs and an extensive set of heterogenous datasets from various data spaces. The HDA provides a single, federated entry point to be consumed by user, DESP components or other DEDL services. Data discovery is powered by the STAC specifications [8], with the objective to simplify the

doi:10.2760/46796

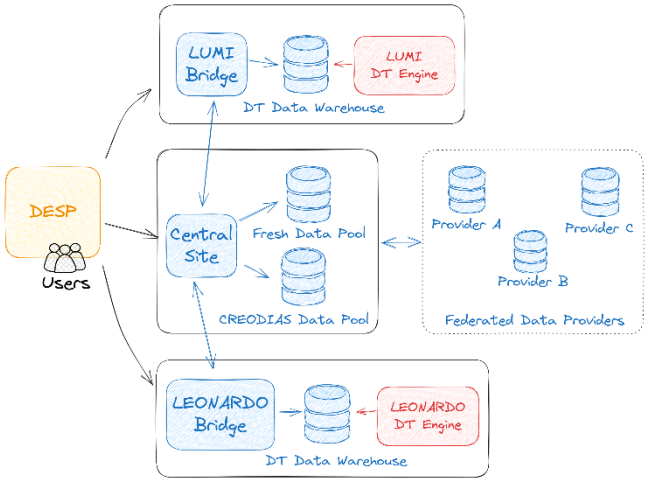

Figure 1: Conceptual overview of DestinE with focus on DEDL components [blue]. Components of DTE are depicted in red and DESP in yellow.

discovery and handling of data. DEDL offers a single data catalogue exposed via the STAC API backed by the software of EODAG (Earth Observation Data Access Gateway) [4]. EODAG is a Python package for searching, aggregating results, and downloading data via a unified API for data access regardless of the data provider or location. Hence, the HDA API makes downloading of geographically distributed datasets into a central data repository obsolete. Instead, datasets can be directly accessed at the source via the inherent data access protocols. A so-called Fresh Data Pool, see Figure 1, is implemented to act as a caching layer to optimise data access via the HDA in a transparent way. The Fresh Data Pool incorporates the CREODIAS Data Pool, hosting all Copernicus Sentinel data. Furthermore, cloud infrastructures at each location in the DEDL expose a set of big data processing services to interact with the available data via the HDA API, with the objective to reduce or mitigate data transfer to a centralised location. DEDL differentiates three categories of big data processing services, referred to as Islet, Hook, and Stack services, each providing specific capabilities to the DEDL.

The Islet service offers direct interaction with the cloud infrastructure deployed at the given locations, providing access to compute, storage, and networking capabilities (Infrastructure as a Service). Furthermore, users can create their own Platform as a Service (Paas) Layer, by deploying their own Kubernetes clusters to run container workloads of their choice. The Hook and Stack service are built on top of the Islet service, fully utilising the PaaS layer to run workloads in Kubernetes.

Pre-configured and ready to use Functions as a Service (FaaS) are made available via the Hook service. It provides a way to enable serverless computing, running pre-defined or user-defined workflows. Two REST API specifications are foreseen to be implemented to interact with the Hook service. Initially, the Hook service can be consumed via the ESA ODPRIP specification representing an OData API [13]. Finally, the DEDL project plan foresees the implementation of the openEO API [12] specification to enhance the service offering.

The Stack service portfolio mainly focuses on the Jupyter software ecosystem, and the Python based parallel computing library Dask [2]. All components of the Stack service are deployed in a fully scalable, multi-tenant and ready-to-use environment individually on each DEDL cloud infrastructure location. The DEDL service architecture is designed to facilitate processing close to the data, referred to as data proximate computation. Enabling this concept will ensure to minimise data transfer between the various locations across Europe and enable scalability of workflows. The deployed Stack services JupyterHub, Dask and Jupyter Kernel Gateway are configured and implemented to meet this requirement. These service components have been selected because of their wide adoption and support given by the Pangeo community [9]. Data proximate computation is enabled by individually spawning Jupyter Kernels or the creation of Dask Clusters at the different locations within DEDL. The Stack service JupyterHub provides a web-based interface to facilitate the use of these cloud computing resources.

3. DASK FOR DATA PROXIMATE COMPUTATION

The Stack service Dask is a fully managed service enabling access to large-scale cloud compute resources, directly from within Python. Users can bring, develop, and run their own code without the need of expert knowledge on how to deploy applications in cloud environments.

3.1. Service deployment considerations

The Dask ecosystem provides various options to deploy and scale Dask workloads on remote clusters such as dask-jobqueue [5], dask-kubernetes [6] or dask-yarn [7]. These deployments rely on a shared network setup for data exchange between a Dask scheduler instance and worker nodes performing the computations. Spanning such clusters across multiple cloud environments is challenging and requires considerable effort and networking expertise. Complexity can be reduced by setting up a dedicated virtual private network (VPN) across the different locations, connecting cloud resources in a single network as shown by [3]. Creating such an IPv6 network mesh (VPN) ensures a high scalability of Dask clusters, however, lacks the capability of multi-tenancy. Hosting multiple users would require the creation of multiple VPNs, one per user, across the given locations. In addition, access to the cluster is limited to environments within the VPN exclusively. Remote access to such Dask clusters is only available via dedicated login nodes or joining the VPN itself.

doi:10.2760/46796

Multi-cloud Dask environments based on VPNs will become a maintenance burden with an increasing number of users as anticipated by the DEDL.

Accordingly, the DEDL Stack service Dask makes use of a different deployment approach utilising the Dask Gateway [9] project. Dask Gateway provides a secure and multi-tenant server for managing Dask clusters. One dedicated Dask Gateway instance is deployed for each location in the DEDL multi-cloud environment as shown in Figure 2. This deployment option allows for various usage scenarios required by the DEDL. All Dask clusters are provisioned in dedicated Kubernetes cluster per DEDL location, without the need of direct access nor knowledge about Kubernetes itself. Furthermore, these Dask clusters are centrally managed enabling the user to create and destroy cluster as needed and as such can be seen as ephemeral compute resources. A set of so-called cluster options are available to the user to customise the environment used by Dask and to vertically scale the utilised Dask workers within a pre-defined range.

3.2. Multi-cloud processing with Dask

Utilising the Dask Gateway Python library [9] a user can connect to Dask Gateway server requesting a cluster. Interaction with the Dask Cluster or more specifically with the scheduler is provided via a the dask.distributed.Client object in Python [15]. For each DEDL location a dask.distributed.Client object can be retrieved to interact with each location separately. A proof of concept of this was presented in [10], showcasing data proximate computation in a multi-cloud environment across Amazon Web Services (AWS) and Google Cloud Platform (GCP).

This concept was taken further during the DEDL project implementation. The basic idea is to utilise metadata attributes provided by the DEDL data discovery service to route computation of the given STAC collection to the corresponding Dask cluster, running at the location of data. Therefore, an abstraction layer is introduced, the DEDL Stack Python client library [16], supporting the user to manage Dask clusters across the DEDL multi-cloud environment. The DEDL Stack client simply wraps the API of the Dask Gateway Client [9] to enable the user to manage multiple Dask Clusters, as if they would manage a single cluster. It is important to note, that those individual Dask clusters are completely isolated from each other, not sharing a private network connectivity or other resources across them. Therefore, direct data exchange between workers on different locations is not possible. In detail, the DEDL data discovery service exposes the STAC API to query the data catalogue available in DEDL. When loading the data into an in-memory representation, xarray [11] objects backed by Dask, location metadata can be directly annotated on the corresponding xarray object. Utilising the client library to execute the required processing workflow, Dask Task Graph, the location information annotated by the data array will be used to submit the individual processing tasks to the corresponding Dask clusters at the given location. The DEDL Stack client incorporates a simple decision tree if access to multiple locations is required to execute the computation.

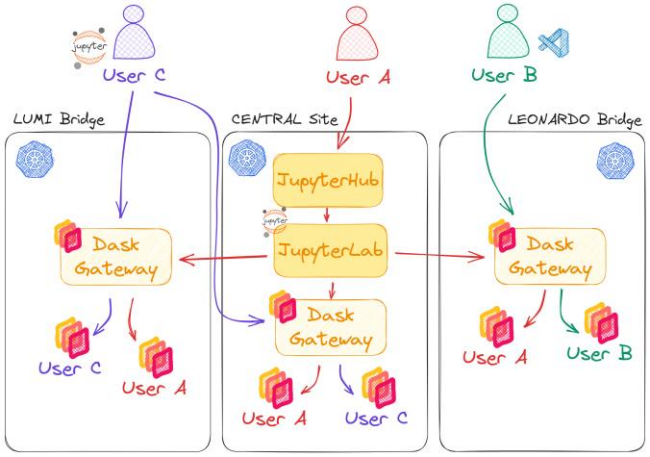

Figure 2: Overview of potential usage scenarios to interact with the DEDL Stack service Dask. Users can simultaneously connect to multiple locations via their development environment of choice.

4. USE CASE DEMONSTRATION AND DISCUSSION

These capabilities are utilised to demonstrate two use cases with a high socio-economic impact. The first use case showcases the potential benefits for disaster management experts to assess flood risks by utilising the DEDL. The use case investigates the 2022 floods in Pakistan, exploiting satellite-based flood data originating from the Global Flood Awareness System (GloFAS) [18], the Global Human Settlement Layer (GHSL) [19] and model data from ERA5-Land [20] to simulate outputs of the Digital Twins. The second use case was developed to support decision makers in assessing the impact of droughts to communal and agricultural water supplies in the example of the 2022 drought in the Po valley in Northern Italy. High resolution soil moisture observations deduced from Sentinel-1 are used together with DTE outputs simulated by ERA5-Land and Corine Land Cover data to investigate the potential impact. Data used in both use cases was pre-processed with the aim to show the advantage of cloud optimised data formats and a common spatial reference system as provided by DEDL data cube service, which is not available within this phase of the project. The use cases have been realised via Jupyter notebooks [17] allowing for interactive data processing and visualisation of results. For each use case a single notebook was created, analysing the simulated DTE outputs and auxiliary input data to retrieve in-time information for decision making. Data extraction and processing was done via the DEDL Stack client library utilising compute

doi:10.2760/46796

resources at LUMI bridge to analyse the simulated DTE outputs. Earth observation datasets have been independently accessed and processed at the DEDL Central Site. The use cases act as a proof of concept, accordingly the amount of data involved is limited in size (< 200 GB), nevertheless, the dynamic scalability of cloud resources across multiple cloud infrastructures has been confirmed. Within a couple of minutes, the required data analytics is performed, allowing decision makers to deduce the demanded information, and even run various scenarios to further support their decision-making process.

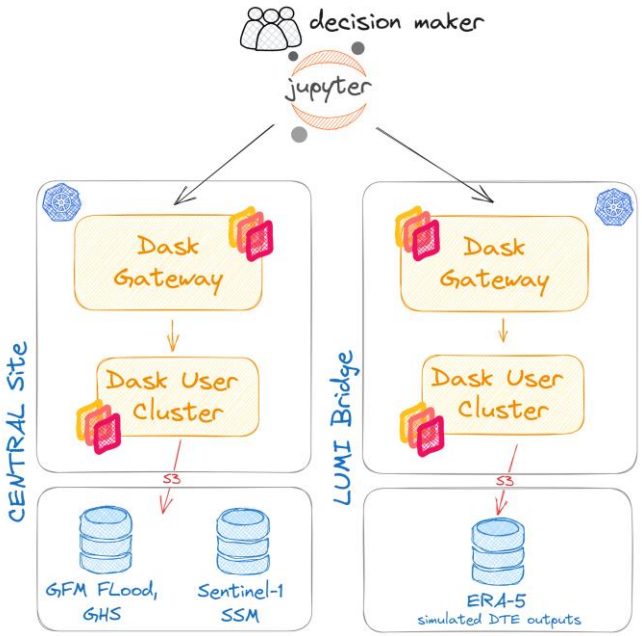

Figure 3: Use case demonstration following the concept of data proximate computation.

5. CONCLUSION AND OUTLOOK

An approach to enable data proximate computation with Dask in the DEDL multi-cloud environment was presented. Utilising metadata attributes to control where data processing is happening might not be an innovation, however, the simplicity of the approach creates future opportunities to evolve the Dask ecosystem. The DEDL Stack client library will be further developed to enhance the user experience and support upcoming usage scenarios of the DESP.

ACKNOWLEDGEMENT

This work has been funded through a contract in the framework of the Destination Earth Data Lake (DEDL) Service issued by EUMETSAT. The authors acknowledge the support, the valuable feedback, and technical recommendations from the entire DEDL Service project team and EUMETSAT.

REFERENCES

[1] Duatis Juarez, J., Schick, M., Puechmaille, D., Stoicescu, M., and Saulyak, B.: Destination Earth Data Lake, EGU General Assembly 2023, Vienna, Austria, 24–28 Apr 2023, EGU23-7177, https://doi.org/10.5194/egusphere-egu23-7177, 2023.

[2] Dask Development Team (2016). Dask: Library for dynamic task scheduling. https://dask.org

[3] Karatosun, A., Grant, M., Baousis, V., McGregor, D., Care, R., Nolan, J., and Tervo, R.: Data Proximate Computation; Multi-cloud approach on European Weather Cloud and Amazon Web Services, EGU General Assembly 2023, Vienna, Austria, 24–28 Apr 2023, EGU23-3639, https://doi.org/10.5194/egusphere-egu23-3639, 2023.

[4] EODAG, created by CS GROUP – France, https://eodag.readthedocs.io/en/stable/index.html#

[5] Dask-Jobqueue, https://jobqueue.dask.org/en/latest/

[6] Dask Kubernetes, https://kubernetes.dask.org/en/latest/

[7] Dask Yarn, https://yarn.dask.org/en/latest/

[8] STAC Specifications, https://stacspec.org/en

[8] Pangeo Community Website, https://pangeo.io/index.html

[9] Dask Gateway, https://gateway.dask.org/

[10] Tom Augspurger: Multi-Cloud workflows with Pangeo and Dask Gateway, https://github.com/pangeo-data/multicloud-demo/tree/master#multi-cloud-workflows-with-pangeo-and-dask-gateway

[11] xarray, https://docs.xarray.dev/en/stable/index.html

[12] openEO API Specification, https://api.openeo.org/

[13] Hook Service: OnDemand Processing API OData v1 specification, https://odp.dataspace.copernicus.eu/odata/docs

[14] EuroHPC Joint Undertaking, https://eurohpc-ju.europa.eu/index_en

[15] Dask.distributed, https://distributed.dask.org/en/stable/

[16] DEDL Stack Client source code, https://github.com/eodcgmbh/dedl-stack-client

[17] DEDL use case example notebooks, https://github.com/eodcgmbh/DEDL-Demonstrator

[18] Global Flood Monitoring System, https://www.globalfloods.eu/

[19] Global Human Settlement Layer, https://ghsl.jrc.ec.europa.eu/

[20] ERA-5 Land, https://www.ecmwf.int/en/era5-land