
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2023 SCPE. Volume 24, Issues 1, pp. 35–44, DOI 10.12694/scpe.v24i1.2041

SCALABLE DATA PROCESSING PLATFORM FOR EARTH OBSERVATION DATA REPOSITORIES

HRACHYA ASTSATRYAN*, ARTHUR LALAYAN† AND GREGORY GIULIANIINSTITUTE FOR ENVIRONMENTAL SCIENCES, UNIVERSITY OF GENEVA, GENEVA, 1205, CHÂTELAINE, SWITZERLAND (GREGORY.GIULIANI@UNIGE.CH)

Abstract. Earth observation (EO) satellite data is essential to environmental monitoring. At a national and regional level, the open data cubes harness the power of satellite data by providing application programming interfaces and services to the end-users. The volume and the complexity of satellite observations are increasing, demanding novel approaches for data storing, managing, and processing. High-performance computing (HPC) and cloud platforms may improve Big EO data processing performance. However, it is necessary to consider several vital aspects for efficient and flexible EO data processing, such as the interoperability from cloud-HPC and EO data repositories, automatic provisioning and scaling of cloud-HPC resources, cost-effectiveness, support of new EO data formats and open-source packages, or linkage with data cube platforms. The article proposes a scalable EO data processing platform interoperable from cloud-HPC and EO data repositories. The platform enables linking any data repository supporting web coverage service or SpatioTemporal Asset Catalog Application Programming Interfaces (STAC-API), and any cloud or HPC resource supporting scheduling system API for providing access to the cluster backends.

Key words: Earth observation satellite data, cloud and HPC, DataCube, STAC, COG, Dask

AMS subject classifications. 68T09

1. Introduction. Earth Observation (EO) satellite data is a crucial enabler for facilitating environmental monitoring [1]. The growing volume and complexity of EO data are making it harder to store, manage, and process [2]. The Earth Observation Data Cube (EODC) paradigm has been proposed to overcome the challenges associated with EO Big Data [3]. The Open Data Cube (ODC) is a well-known implementation of EODC helping scientists work with Analysis Ready Data (ARD) [4]. The ODC focuses on real environmental issues [23] instead of managing and solving EO Big Data challenges. National ODC systems have already been successfully implemented and provided by several countries [5] to monitor, detect and potentially resolve various types of environmental problems, like Australia [6], Switzerland [23], Brazil [22], or Armenia [7].

As a rich open-source environment harnessing satellite technology and EO data, the ODC provides communities with a diverse portfolio of advanced services and tools to make it easier to work with EO data. The essential components of ODC [9] are the following:

- datacube-core - a data analysis framework for data indexing, searching and ingesting.
- datacube-ows - interoperable access to indexed data via several Open Geospatial Consortium (OGC) web services (Web Map Service, Web Map Tile Service, Web Coverage Service).
- datacube-stats - application to calculate large-scale temporal statistics for processing the data indexed in the ODC catalog.
- datacube-explorer - software application responsible for the visualization and analysis.

Besides the enumerated benefits, ODC has some limitations and difficulties in use. Though it provides scalable computational opportunities and optimal data processing and fetching, it is hard to provision third-party computational infrastructures with ODC automatically. Newly implemented tools and platforms address the issues of optimal EO data processing and extracting in cloud-based virtualized environments. In recent years, researchers have used widely new methods to store, fast search, fetch and process the data optimally, such

*Institute for Informatics and Automation Problems National Academy of Sciences of Armenia 1, Paruyr Sevak str. 0014 Yerevan, Armenia (hrach@sci.am)

†Institute for Informatics and Automation Problems National Academy of Sciences of Armenia 1, Paruyr Sevak str. 0014 Yerevan, Armenia (arthurlalayan97@gmail.com)

as Cloud Optimized GeoTIFF (COG) [3] and the SpatioTemporal Asset Catalog (STAC) [4]. COG is a tile-based image format to support compression for reducing the transmitted data and the data transfer time. It allows users to request part of the file using an HTTP range request [3]. The STAC [8] provides particular metadata (in JSON or GeoJSON formats) about the remote sensing image to facilitate indexation and querying of EO data. Working with COGs and STAC metadata, it is evaded loading the corresponding image in memory to request, search or find an exciting part of the data. The search and query are organized in a flexible way on the level of metadata rather than the image having a bigger file size. The query finds data with the corresponding request, loads the part of the exciting area (or a particular band) from the EO data into memory, and proceeds with data processing. There are already public EO datasets with the STAC API, such as the Sentinel-2 Cloud Optimized GeoTIFFs on Amazon Web Services [6].

Since the storage, processing, and management of EO data is a complex task, several mentioned methods and tools need to be considered to overcome the hurdles associated with EO data by making data storage and processing more sensitive. Still, remote sensing data is growing daily, and scalable data processing is a real challenge [2]. Thus, there is a need to overcome the data processing obstacle, and distributed frameworks, such as Apache Spark or Dask (both with master-slave architecture), can cope with the Big EO data processing by providing scalability and efficiency [25].

The article proposes a scalable EO data processing platform interoperable from cloud-HPC and EO data repositories. It extends the functionality of ODC by bringing it to third-party computational infrastructures and providing optimized solutions. The platform enables linking any data repository supporting web coverage service or any cloud or HPC resource supporting Dask gateways.

The structure of the paper is the following. Section 2 (Motivation) describes the motivation of the work, and Section 3 (Related work) reviews several EO data processing techniques. Section 4 presents the scalable platform, Section 5 evaluates the platform, and Section 6 concludes the article.

2. Motivation. Increasing EO data requires enormous computing resources to process satellite remote sensing data. The research communities and end-users set up private HPC and cloud infrastructures or use the resources of global cloud providers to address the increasing needs of EO data processing, which becomes more complex and requires more hardware resources and flexible software. The following key performance indicators (KPI) are critical for the efficient processing of EO data:

1. **Performance and scalability.** Shared memory and distributed memory techniques and tools increase the application's performance and scalability. These techniques process EO data in parallel to reduce the execution time of the processing, as the area of interest or time-series study may enlarge the size of the processing data making the processing time longer.
2. **Interoperability from cloud-HPC and EO data repositories.** Interoperability of the EO data processing platform from the data and computing infrastructures gives several benefits. Firstly, separation from the computing infrastructure allows deploying the service using private or public HPC and cloud infrastructures. Secondly, by separating the processing platform from the data repository, the platform will become operable to extract the data from any remote sensing data repository.
3. **Automatic and fast cloud-HPC provisioning and scaling.** The predefined automated tools allow the processing of the request according to the data size without manual intervention. If the number of scaled processes is kept constant, many resources, such as CPU or energy, will be wasted even in the idle state. Therefore, it is essential to scale the processing on demand.
4. **Cost efficiency.** Using open-source solutions and deployments increases the cost-effectiveness of the processing platform. It is also essential to consider the flexibility in later updates.
5. **Support of new formats and novel solutions.** It is always essential to follow novel solutions and remote sensing image formats to make storing, fetching, or managing the data more efficient.
6. **Linkage with DC.** The EO data processing platform linkage with the Data Cubes (DC) is critical, as DC provides many vital tools and services to index and ingest data. Moreover, it provides the implementations of the OGC services, such as WMS, WCS, or WMTS.

[3] http://www.cogeo.org/
[4] http://stacspec.org/
[6] https://registry.opendata.aws/sentinel-2-l2a-cogs/

The ODC deployment on a single server is simple compared to the linkage with any HPC cloud infrastructure, considering KPIs. However, creating a scalable processing platform considering all the mentioned KPIs is challenging, as it needs to combine different architectures efficiently. Therefore, the national DCs (Australia, Armenia, Switzerland, or Brazil) rely on customized solutions using HPC infrastructures.

3. Related work. EO provides essential data for environmental monitoring, whereas the ODC offers various tools and services for processing remote sensing data and extracting useful information. Several research works discuss the use cases of ODC in monitoring different kinds of environmental issues on a local or global scale. The focus of this section is to examine the novel services and their limitations considering defined KPIs.

Many research projects utilize the EO satellite data for environmental monitoring and assessment, such as creating an analytical platform for weather data visualization [28] or crop yield estimation [29] using satellite image processing. Nevertheless, in the research works, the processing of EO data is carried out without considering the data processing performance considering novel data formats, solutions, and scalability. In addition, the absence of the DC is noticed, as this could facilitate the work of collecting data, processing, and visualizing them in both cases, therefore, none of the KPIs is considered.

The ODC services primarily rely on the computational facilities of the ODC server, where the actual computations are carried out. Swiss DC uses ODC [10] for efficient EO data analysis to monitor land degradation on a national scale, while [11] for forest monitoring of eastern Taiwan. The African ODC [12] provides a portfolio of services, such as crop phenology monitoring in Ghana, forest monitoring in Senegal, mangroves in Sierra Leone, and land degradation monitoring in Tanzania. The researchers [13] use ODC to monitor changes such as deforestation, urbanization, and coast evolution. Similar services are available in the Armenian ODC, which provides the air temperature forecast service validated for the Ararat Valley [16] and a shoreline delineation service with the case study for the Lake Sevan [24]. The works mentioned above use the ODC without considering KPIs 1, 2, and 3.

The authors [14] suggest a new DC on Demand approach to overcome the complexities in customizing the ODC per the user demand by generating an ODC instance virtually to address the user request (particular area, satellite type, and the observation time). However, the users may receive scalable computational resources per request by scaling the DC instance per user demand without considering KPIs 1 and 2.

The utilization of distributed computing frameworks is a natural choice to overcome the ODC scalability limitation [25, 15]. For instance, a pure python framework Dask enables running the Pandas or NumPy data frames locally or on a computational cluster relying on the master-slave architecture. The authors [25] use ODC and Dask to create a spatial data analysis tool for the scalable processing of EO data. The authors use a fixed number of worker nodes rather than on-demand worker node scaling without considering KPI 3.

The usage of Big Data frameworks, such as Apache Hadoop or Spark, is also quite promising and widely implemented, as the frameworks can provide efficient and distributed environments for big data processing [17]. The paper [18] shows the effectiveness of an adaptive Spark-based remote sensing data on-demand processing method on the cloud, with the data storage Hadoop Distributed File System (HDFS), which is more efficient in means of execution time than Hadoop. This method has limitations, as virtual machines are used instead of lightweight containerized images. Container-based virtualization saves resources and reduces the overhead of hypervisor-based virtualization; hence, the containers are faster than the virtual machines. Therefore, containers instead of virtual machines may reduce the necessary resources and provide faster scaling. Furthermore, writing code in Spark for remote sensing data processing is complex without using already ready-made and open-source libraries. Besides, Spark requires adjusting the programming code with the environment. The papers [19, 20] use Spark-on-Kubernetes deployed on a cloud to overcome the mentioned issues. In [19], researchers suggest a task scheduling mechanism to change the worker pods dynamically. GeoPySpark, based on Spark, provides a scalable remote sensing data processing system, where HDFS is deployed on nodes of Kubernetes to diminish data transfer between workers [20]. The limitation of the latest three works is the absence of linkage with DC, not interoperable from data repositories, as they either use HDFS or external storage. Hence, they can only process data from remote sources if they consider KPIs 2 and 6.

Serverless platforms provide only the code for EO data processing and transfer control of the server to the service provider. The Amazon Web Services (AWS) Lambda-based platform processes remote sensing images on a serverless platform [21] by splitting the image into small tiles and simulating using a Lambda function.

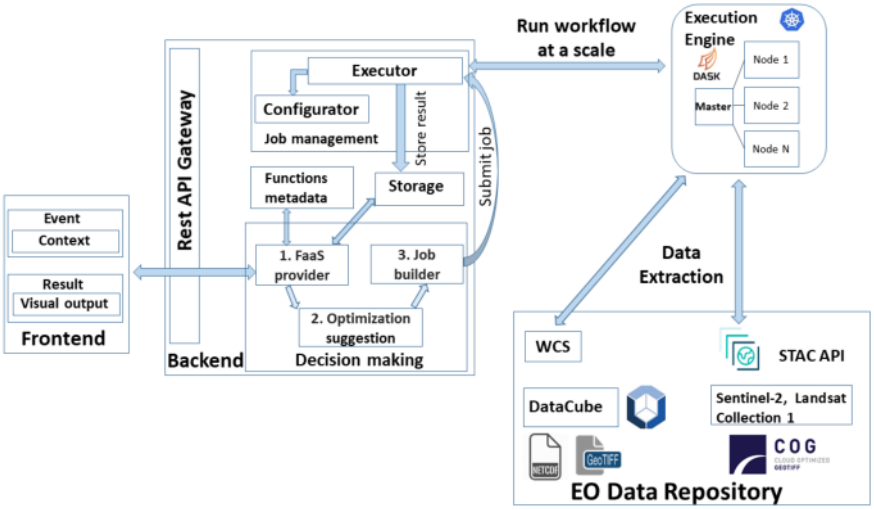

Fig. 4.1: The structure of the scalable data processing platform.

The authors also show that a novel Function as a service (FaaS) approach is faster than Spark and Ray. The limitation of this approach is that it depends on AWS lambda (KPI 2). Moreover, all implementations of FaaS will hinge on some product or cloud provider. Therefore, this work did not consider KPIs 2, 4, and 6. Besides the mentioned limitations, all mentioned works did not consider novel data formats and solutions (KPI 5). The paper aims to suggest a scalable platform to consider all mentioned limitations.

4. Architecture. As shown in Fig. 4.1, a novel scalable platform [9] is suggested to satisfy the KPI requirements, consisting of Frontend, Backend, Execution Engine, and Data Repository modules.

The suggested scalable EO data processing platform consisting of several layers hides the complexity of EO data storing, processing, and visualizing from a user. Performance and scalability (KPI 1) is obtained through the Execution Engine module, which can use any cloud or HPC resources supporting Dask gateways. Performance improvements are achieved by scaling the number of compute nodes for a Dask cluster and distributing data processing between the nodes. The data repositories and computational resources are separated to provide interoperability from cloud-HPC and EO data repositories (KPI 2). The platform is implemented using open-source solutions and deployments, considering new formats (COGs) and novel STAC API solution (KPI 5), and the possibility of fetching the data from the DC (KPI 6) using the WCS service to ensure the cost-efficiency (KPI 4). The optimization suggestion module ensures automatic and fast cloud-HPC provisioning and scaling (KPI 3), providing computing resources on demand based on the size of the input data.

The workflow is as follows: a user request to the Backend using Frontend, and the decision-making component first does its job, then it refers to the job management module, which in turn refers to the Execution Engine module, and the latter fetches data from Data Repositories, processes according to the request, stores the result and sends back as the response to the user.

4.1. Frontend. The Frontend module provides a user interface to interact with the Backend smoothly. The module supports asynchronous requests to submit event-driven workflows without waiting to complete existing ones. It offers quick and easy access to the resources by hiding the complexity of EO data processing and visualization from a user. The geoprocessing job is directly submitted to the scalable framework via a rich web-based interface. The Frontend module gets the result and visualizes it on a map. The Frontend query contains the area of interest, period, and function with its arguments in the request body. After the selection,

[9] https://github.com/arthurlalayan/eo-platform

Table 4.1: Parameters of a request body.

Request parameter	Description
Function name	A function that will be processed
Area	FeatureCollection in JSON format. It is a polygon containing the coordinates of the polygon points.
Optional arguments	Optional function arguments, such as the coordinate system (Geodetic Parameter Dataset:4326) for projecting or the data repository name from which the data will be extracted.

the Frontend calls the Backend using the POST method of the Hypertext Transfer Protocol (HTTP) and waits for the response. The information of a request body is shown in Table 4.1.

The users select an area of interest by drawing a polygon on the map. Then, the area is converted into a FeatureCollection object in the JSON format considering the OGC standard. The Backend module returns the uniform resource locator (URL) to download or visualize if the processing result is an image. In the visualization case, the WMS or WMTS services are accessible through the datacube-ows package. The dynamic tiling provided by titiler[7] optimizes large-scale visualization workflows by creating a tile server that can access raw data, scale, reproject, or encode the data into an image.

4.2. Backend. The Backend provides RESTful (Representational State Transfer) API to process the Frontend requests. It consists of decision-making and job management components.

The FaaS provider module of the decision-making component first checks whether the system supports the user's request and the function metadata component contains supported functions, processing predefined types, and the code. The module examines if the request has been processed in the past and the result will be retrieved without processing if it has already been processed. The checking functionality is based on the storage, which couples and stores request data and the processing results. The data will be processed by considering all tiles that match the user request area, and the last check will be considered if the tile was processed in the past.

Then, the complexity of the data processing is evaluated to determine the characteristics of the computational resources. The optimization module provides the number of computational nodes considering the input data size and the chunk size affecting the simulation speedup. The module relies on the adaptive scaling of Dask to balance the performance and resources by receiving a minimum and a maximum number of worker nodes as input. The minimum and the maximum numbers of worker nodes are determined considering the random-access memory (RAM) sizes of Dask nodes. Any heuristic and machine learning methods (regressions, neural networks) or multi-object optimization algorithms can be easily linked to the module to optimize performance, cost, or power consumption. The input data size is divided by the available RAM size of the worker node to calculate the minimum. At the same time, the maximum is equal to the minimum multiplied by 1.5. Rounding up provides an integer value for the maximum and minimum. For instance, in the case of the 64 GB input data size and 4 GB RAM per Dask worker node, the minimum is 64 / 4 = 16, while the maximum is 24 (16*1.5).

Finally, the job builder prepares and submits it to the job management module. The job contains information on the request body, chunk size, and the number of computing nodes provided by the optimization suggestion module. The chunk size parameter is set to auto, which means Dask will divide the data into optimal chunks.

The job management module consists of executor and configurator components. The executor receives the jobs, accesses the computing module for processing, executes the function, gets the result back, and stores it in the storage. If the computing resources are unavailable, the executor keeps jobs in the queue and executes them when resources become available. The configurator is based on the configuration files to configure the URLs and credentials of the remote Execution Engine and the Data Repositories. Therefore, it provides the

[7]https://developmentseed.org/titiler/

Table 4.2: Configuration parameters.

Request parameter	Description
URL of the Execution Engine module	The URL of a Dask Gateway
Credential	Encrypted credential for the Dask gateway
Data repositories	Contains the list of data repositories in which each repository has a name, URL and type (either WCS or STAC API)

flexibility to connect a remotely accessible Execution Engine module and later extract data from different data repositories. The configuration files are in JSON format and contain the information provided in Table 4.2.

4.3. Execution Engine. The Execution Engine module provides the scalability and the performance of the EO data processing. The module processes data in parallel using an HPC cluster and automatically scales the computing nodes on demand. A Dask framework [26] has been selected for distributed computing because it is an open-source Python library. The Dask is an easy-to-use library without configuration, setup, or complete code changes. It is possible to create a Dask cluster that can be deployed and scaled on the cloud or HPC. As the Execution Engine module is separated from the other modules, it is necessary to use, manage or scale it remotely. Dask Gateway [8] provides a secure and multi-user server for managing Dask clusters remotely.

Moreover, it can be installed on Hadoop and Kubernetes clusters or an HPC Job Queue. Hence, it is possible to connect to the Execution Engine module, scale on-demand, extract data, and execute a function on a Dask cluster parallel. Gateway provides a python API for connecting to the Dask remote cluster, and it can easily be deployed on a Kubernetes cluster, similar to other services. After the deployment, there is a need to use Gateway's python API to connect to the Gateway by providing a public host or IP, port, and authentication type and credentials. The Kerberos authentication method is chosen for platform security. After connecting to the Gateway, it is possible to create a Dask cluster by providing the number of nodes and their characteristics (such as CPU and memory). The cluster with the demanded resources and nodes will be created where the Gateway is deployed (e.g., in Kubernetes, each node will be a pod with the provided specifications). We consider Kubernetes as a Dask cluster Backend.

4.4. Data repositories. Data repositories are the storage that stores the remote sensing data. An essential goal of the repository is the availability of remote data to provide interoperability from the data repositories, which makes the platform flexible, as it can work with different data storage. A new solution for providing data using the STAC API and DC will be considered. STAC API provides a REST API to call and query EO data using only lightweight metadata in JSON format and then download it to process. In the case of the DC, there is a need to use WCS, the implementation of which has ODC, to make data open on the Internet. Therefore, the suggested platform is flexible and can be used with WCS and STAC API. The recommended platform considers Armenian DC [7] with the WCS on top of it and Sentinel-2 Cloud-Optimized GeoTIFFs [6] as an example of data repositories. The latest provides satellite images from Sentinel-2 in COG format with STAC API. The data from Armenian DC stored from several satellites, such as Landsat and Sentinel, is being used for monitoring various environmental problems [7, 16, 24]. The WCS is installed on the top of the Armenian DC using the datacube-ows package.

5. Platform evaluation. As a web application, the Frontend module provides a flexible user interface based on React JS framework and the open-source Leaflet library. The module makes it possible to process, analyze and visualize EO data by hiding the complexity of EO data processing. The workflow begins by selecting an exciting area as a rectangle on a map, selecting the period, and an already implemented function to process the EO data (see Fig. 5.1).

In the next stage, the Frontend interacts with the Backend using the Rest API, a middleware between the Frontend and the Backend. The platform's effectiveness is demonstrated by considering the Normalized

[8] https://gateway.dask.org/

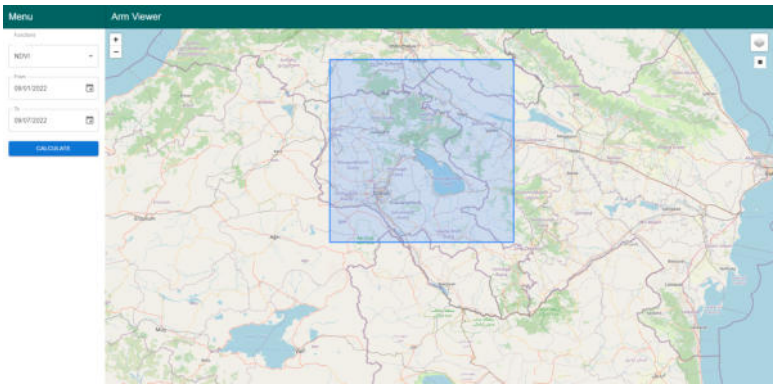

Fig. 5.1: Selecting an interesting area, period, and function in Frontend.

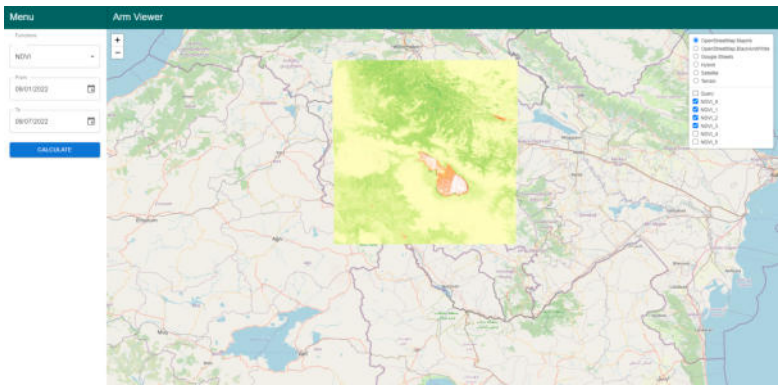

Fig. 5.2: NDVI result visualized on the map.

Difference Vegetation Index (NDVI) as an EO data processing function, which is a graphical indicator for determining the green density of land [30]. The analysis of the index consists of matrix operations, and the formula of the NDVI is the following:

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{5.1}$$

where RED and NIR are the red and near-infrared bands correspondingly extracted by pixels from satellite images received from the European optical imaging Sentinel-2 satellite. As the platform supports new formats and novel solutions, the data can be fetched in COG formats with STAC API or netCDF and GeoTIFF formats from DCs. After the processing, the Frontend module visualizes the output on an interactive map. Fig. 5.2 shows the mean NDVI calculation of some territory of Armenia visualized on a map. The NDVI function results range from -1 to 1, where green corresponds to good vegetation conditions (values around 1) and red to the opposite (-1).

Several experiments have been carried out to evaluate the processing speedup using the scaling and flexible functionality of the Backend module. For example, the average weekly NDVI for the territory of Armenia is calculated, considering 16 GB, 32 GB, and 64 GB of the input data. The resources of the CloudLab [27] have been used for the experiments using the various configurations of Dask clusters deployed on the Kubernetes system. In this particular experiment, each worker node of the Dask cluster corresponds to a pod with a fixed size of resources, 2 cores, and 4 GB RAM on Kubernetes. During experiments 1, 2, 4, 8, 16, and 32 number of Dask worker nodes is considered to evaluate the performance improvement in the distribution of computing.

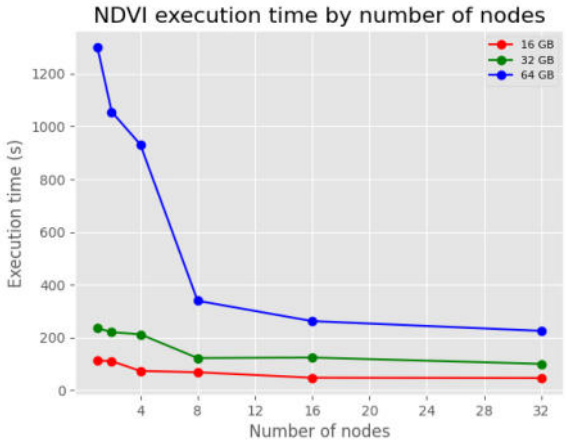

Fig. 5.3: NDVI execution times.

Fig. 5.3 shows the execution time of the NDVI with the specified number of compute nodes for the specified input sizes.

The figure shows that the EO data processing performance is boosted using the scaling of the computational resources in Dask. Scaling the number of workers from 1 to 32 improves performance by 2.44, 2.36, and 5.77 times for the 16, 32, and 64 GB of input data. The experiments show exciting behavior, as the execution time of the NDVI workflow using the optimization module show near execution times compared with the 32 nodes configuration. The optimal suggested configuration provided by the optimization service is 42%, 36%, and 22% behind compared to the run time for 16, 32, and 64 GBs of input data using 32 nodes. Increasing the input data size leads to an increase in the accuracy of the optimization module.

6. Conclusion. A scalable data processing platform has been presented for Earth observation data repositories satisfying several critical KPIs for flexible and optimal processing. The platform improves the EO data processing performance by scaling the processing on computational resources using either cloud or HPC. The platform's primary focus is interoperability from cloud-HPC and EO data repositories, enabling the use of any repository supporting WCS or STAC APIs or any cloud or HPC resource supporting Dask gateways. The platform relies on open-source solutions to provide cost-efficiency and is linked with the DC. In the future, it is planned to enrich the platform by considering various necessary functionality, evaluate the platform with different examples, and use the platform for past resolved issues with Armenian DC to boost performance. The optimal way of storing the satellite images will be studied by using compression techniques to find a trade-off between the data size, the transfer time, and the performance.

Acknowledgments. The research was supported by the University of Geneva Leading House and the State Committee of Science of the Republic of Armenia by the projects entitled "ADC4SD: Armenian Data Cube for Sustainable Development", "Self-organized Swarm of UAVs Smart cloud Platform Equipped with Multi-agent Algorithms and Systems" (Nr. 21AG-1B052) and "Remote sensing data processing methods using neural networks and deep learning to predict changes in weather phenomena" (Nr. 21SC-BRFFR-1B009).

REFERENCES

[1] Giuliani G., Egger E., Italiano J., *Essential Variables for Environmental Monitoring: What Are the Possible Contributions of Earth Observation Data Cubes?*, Data: Vol. 5, 2020, No. 4, doi: 10.3390/data5040100.

[2] Ma Y., Wu H., Wang L., *Remote sensing big data computing: Challenges and opportunities.*, Future Generation Computer Systems: Vol. 51, 2015, pp. 47-60, doi: 10.1016/j.future.2014.10.029.

[3] Giuliani G., Masó J., Mazzetti P., *Paving the Way to Increased Interoperability of Earth Observations Data Cubes.*, Data: Vol. 4, 2019, No. 113, doi: 10.3390/data4030113.

[4] Giuliani G., Chatenoux B., Bono A., *Building an Earth Observations Data Cube: lessons learned from the Swiss Data Cube (SDC) on generating Analysis Ready Data (ARD).*, Big Earth Data: Vol. 1, 2017, No. 1-2, pp. 100-117, doi: 10.1080/20964471.2017.1398903.

[5] Killough B., *Overview of the Open Data Cube Initiative.*, IGARSS 2018 - 2018 IEEE International Geoscience And Remote Sensing Symposium: 2018, pp. 8629-8632, doi: 10.1109/IGARSS.2018.8517694.

[6] Lewis A., Oliver S., Lymburner L., *The Australian Geoscience Data Cube — Foundations and lessons learned.*, Remote Sensing Of Environment: Vol. 202, 2017, pp. 276-292, doi: 10.1016/j.rse.2017.03.015.

[7] Asmaryan S., Muradyan V., Tepanosyan G., *Paving the Way towards an Armenian Data Cube.*, Data: Vol. 4, 2019, No. 3, doi: 10.3390/data4030117.

[8] Zhao Y., Yang X., Vatsavai R., *A Scalable System for Searching Large-scale Multi-sensor Remote Sensing Image Collections.*, In 2021 IEEE International Conference On Big Data (Big Data): pp. 3780-3783, doi: 10.1109/BigData52589.2021.9671679.

[9] Killough B., Siqueira A., Dyke G., *Advancements in the Open Data Cube and Analysis Ready Data — Past, Present and Future.*, In IGARSS 2020 - 2020 IEEE International Geoscience And Remote Sensing Symposium: pp. 3373-3375, doi: 10.1109/IGARSS39084.2020.9324712.

[10] Giuliani G., Chatenoux B., Benvenuti A., *Monitoring land degradation at national level using satellite Earth Observation time-series data to support SDG15 – exploring the potential of data cube.*, Big Earth Data: Vol. 4, 2020, No. 1, pp. 3-22, doi: 10.1080/20964471.2020.1711633.

[11] Cheng M., Chiou C., Chen B., *Open Data Cube (ODC) in Taiwan: The Initiative and Protocol Development.*, In IGARSS 2019 - 2019 IEEE International Geoscience And Remote Sensing Symposium: pp. 5654-5657, doi: 10.1109/IGARSS.2019.8898576.

[12] Mubea K., Killough B., Seidu O., *Africa Regional Data Cube (ARDC) is Helping Countries in Africa Report on the Sustainable Development Goals (SDGS).*, In IGARSS 2020 - 2020 IEEE International Geoscience And Remote Sensing Symposium: pp. 3379-3382, doi: 10.1109/IGARSS39084.2020.9324156.

[13] Vâjâială-Tomici C., Filip I., Pop F., *Landscape Change Monitoring using Satellite Data and Open Data Cube Platform.*, In 2020 IEEE 16th International Conference On Intelligent Computer Communication And Processing (ICCP): pp. 573-580, doi: 10.1109/ICCP51029.2020.9266254.

[14] Giuliani G., Chatenoux B., Piller T. , *Data Cube on Demand (DCoD): Generating an earth observation Data Cube anywhere in the world.*, International Journal Of Applied Earth Observation And Geoinformation: Vol. 87, 2020, pp. 102035, doi: 10.1016/j.jag.2019.102035.

[15] Gomes V., Carlos F., Queiroz G., *Accessing and Processing Brazilian Earth Observation Data Cubes with the Open Data Cube Platform.*, ISPRS Annals Of The Photogrammetry, Remote Sensing And Spatial Information Sciences: Vol. 4, 2021, pp. 153-159, doi: 10.5194/isprs-annals-V-4-2021-153-2021.

[16] Astsatryan H., Grigoryan H., Poghosyan A., *Air temperature forecasting using artificial neural network for Ararat valley.*, Earth Science Informatics: Vol. 14, 2021, doi: 10.1007/s12145-021-00583-9.

[17] Astsatryan H., Lalayan A., *Performance-efficient Recommendation and Prediction Service for Big Data frameworks focusing on Data Compression and In-memory Data Storage Indicators.*, Scalable Computing: Practice and Experience: Vol. 22, 2021, pp. 401-412. doi: 10.12694/scpe.v22i4.1945.

[18] Tan X., Di L., Zhong Y., *Spark-based adaptive Mapreduce data processing method for remote sensing imagery.*, International Journal Of Remote Sensing: Vol. 42, 2021, No. 1, pp. 191-207, doi: 10.1080/01431161.2020.1804087.

[19] Huang W., Zhou J., Zhang D., *On-the-Fly Fusion of Remotely-Sensed Big Data Using an Elastic Computing Paradigm with a Containerized Spark Engine on Kubernetes.*, Sensors: Vol. 21, 2021, No. 9, doi: 10.3390/s21092971.

[20] Guo J., Huang C., Hou J., *A Scalable Computing Resources System for Remote Sensing Big Data Processing Using GeoPySpark Based on Spark on K8s.*, Remote Sensing: Vol. 14, 2022, No. 3, doi: 10.3390/rs14030521.

[21] Yang G., Liu J., Qu M.,, *A Remote Sensing Image Processing System on Serverless Platform.* , In 2021 IEEE 45th Annual Computers, Software, And Applications Conference (COMPSAC): pp. 258-267, doi: 10.1109/COMPSAC51774.2021.00044.

[22] Ferreira K., Queiroz G., Vinhas L., *Earth Observation Data Cubes for Brazil: Requirements, Methodology and Products.*, Remote Sensing: Vol. 12, 2020, No. 24, doi: 10.3390/rs12244033.

[23] Chatenoux B., Richard J., Small D., *The Swiss data cube, analysis ready data archive using earth observations of Switzerland.*, Sci Data: Vol. 8, 2021, doi: 10.1038/s41597-021-01076-6.

[24] Astsatryan H., Grigoryan H., Abrahamyan R., *Shoreline delineation service: using an earth observation data cube and sentinel 2 images for coastal monitoring.*, Earth Science Informatics: 2022, doi: 10.1007/s12145-022-00806-7.

[25] Xu D., Ma Y., Yan J., *Spatial-feature data cube for spatiotemporal remote sensing data processing and analysis.*, Computing: Vol. 102, 2020, doi: doi.org/10.1007/s00607-018-0681-y.

[26] Rocklin M., *Parallel Computation with Blocked algorithms and Task Scheduling..*, 2015.

[27] Duplyakin D., Ricci R., Maricq A., *The Design and Operation of CloudLab.*, In Proceedings Of The USENIX Annual Technical Conference (ATC): 2019, pp. 1-14.

[28] Astsatryan H., Grigoryan H., *Weather Data Visualization and Analytical Platform.*, Scalable Computing: Practice and Experience: Vol. 19, 2018, pp. 79-86. doi: 10.12694/scpe.v19i2.1351.

[29] Anitha M., Priyanka S., *Review of Crop Yield Estimation using Machine Learning and Deep Learning Techniques.*, Scalable Computing: Practice and Experience: Vol. 23, 2022, pp. 59-80. doi: 10.12694/scpe.v23i2.2025.

[30] Huang S., Tang L., *A commentary review on the use of normalized difference vegetation index (NDVI) in the era of popular remote sensing.*, Journal of Forestry Research: Vol. 32, 2021, doi: 10.1007/s11676-020-01155-1.

Edited by: Katarzyna Wasielewska
Received: Aug 4, 2022
Accepted: Apr 4, 2023