Baltic J. Modern Computing, Vol. 11 (2023), No. 3, pp. 420–434
https://doi.org/10.22364/bjmc.2023.11.3.05

A Multi-objective Optimization Service for Enhancing Performance and Cost Efficiency in Earth Observation Data Processing Workflows

Arthur LALAYAN[1,2], Hrachya ASTSATRYAN[1], Gregory GIULIANI[3,4]

[1] Institute for Informatics and Automation Problems National Academy of Sciences of Armenia
[2] National Polytechnic University of Armenia
[3] Institute for Environmental Sciences, University of Geneva, Switzerland
[4] UNEP/GRID Geneva, Switzerland

arthurlalayan97@gmail.com, hrach@sci.am, Gregory.Giuliani@unige.ch

ORCID 0000-0001-6695-3697, ORCID 0000-0001-8872-6620, ORCID 0000-0002-1825-8865

Abstract. Earth observation technology has become increasingly crucial for monitoring various aspects of our planet through systematic data collection. However, processing the large volume of satellite data generated can be challenging, requiring high-performance computing solutions. The Dask framework has gained popularity for its flexibility and efficiency in processing large amounts of data in a distributed manner. Nevertheless, determining the optimal Dask cluster configuration remains challenging, as it requires balancing performance and cost objectives. A novel multi-objective optimization service is proposed to address this challenge that enhances the performance and cost efficiency of earth observation data processing workflows. Our approach is to generate a set of Pareto-optimal solutions, allowing users to make informed decisions regarding the optimal trade-offs between performance and cost. The effectiveness is demonstrated using real-world earth observation datasets and outperforms existing performance and cost-efficiency solutions.

Keywords: Earth observation, HPC over cloud, Dask, multi-objective optimization, Pareto-optimal

1 Introduction

Earth observation (EO) data obtained from satellites play a crucial role in monitoring various layers of the Earth, including the land, atmosphere, and ocean (Camps-Valls et al., 2016). However, the growing volume of EO data has created new challenges in handling and analyzing such large amounts of data (Yao et al., 2019). To overcome these challenges, high-performance computing (HPC) has emerged as an effective solution for improving data processing capabilities by distributing computation across

multiple machines (Li, 2020; Astsatryan et al., 2023). The most commonly used HPC platforms for EO data processing are the open-source parallel Python Dask library (Rocklin, 2015) and the Apache Spark engine (Zaharia et al., 2016), using a master-slave architecture. EO communities widely adopt the Dask library due to its abundance of tools and libraries for handling geospatial data using Python. Dask can be configured and customized in various virtual, physical, cloud-based, and on-premises computing environments, making it a versatile solution for parallelizing EO data processing tasks. The execution time of EO data processing tasks depends on the configuration of the Dask cluster, particularly the number and computational characteristics of worker nodes (central processing unit - CPU and random-access memory - RAM). At the same time, cloud providers offer various instances with varying numbers of CPUs and sizes of RAM, each incurring different costs. Finding the optimal trade-off between performance and cost is challenging when determining the optimal Dask configuration.

On the one hand, utilizing more computational resources can lead to greater parallelization of data processing. On the other hand, this incurs higher costs from cloud providers. The article proposes a multi-objective optimization method for optimal EO data processing that considers performance and cost objectives to address this challenge.

The article is organized as follows: In Section 2 (Problem formulation), we formulate the problem of optimizing EO workflows. Section 3 (Related work) provides a comprehensive review of previous optimization techniques used in the field of EO. Section 4 (Methodology) outlines the methodology of our proposed approach. In Section 5 (Evaluation), we evaluate the effectiveness of our method. Finally, Section 6 (Conclusion) summarizes the essential findings and implications of the study and concludes the article.

2 Problem formulation

HPC over the cloud is a popular and practical alternative to traditional on-premise clusters for running embarrassingly parallel or loosely coupled jobs (Malla and Christensen, 2019). There are several methods of delivering HPC through the cloud, such as virtual machines or containers hosted on the cloud infrastructure. However, the performance of these solutions can vary significantly based on their configuration, making it challenging to determine the optimal balance between performance and cost. We aim to provide users with the best possible performance while minimizing costs. Thus, this article addresses the challenge of finding an optimal balance between performance and cost for EO data processing in the cloud.

Cloud service providers offer various instances with varying quantities of CPU and RAM. Table 1 lists some of the widely used compute-optimized instances provided by Amazon's Elastic Compute (EC2) cloud service.

The table showcases the diverse options for computing instances, ranging from 2 to 64 CPUs, each with a corresponding hourly cost proportional to the number of CPUs. Dask clusters rely on selecting appropriate instance types, which offer varying combinations of CPU, memory, and storage resources, and determine the number of worker nodes that can be launched in the cluster. The challenge lies in finding the optimal

Table 1. List of some compute-optimized EC2 instance types.

Instance name	vCPU	Memory (GiB)	Hourly rate ($)
large	2	4	0.1134
xlarge	4	8	0.2268
2xlarge	8	16	0.4536
4xlarge	16	32	0.9072
8xlarge	32	64	1.8144
16xlarge	64	128	3.6288

instance type and quantity that balances performance and cost-effectiveness. Each instance comes with a cost proportional to the number of CPUs it offers. Moreover, the user must consider several other critical factors, such as the limitation of execution time and budget constraints for the computational resource cost. Besides, the EO data processing time depends on various factors, such as function complexity, input data size, or the type of complexity computational infrastructure.

EO data is acquired by specialized sensors that capture images and various forms of data about the Earth from space. The data obtained can be represented as matrices, making it possible to process EO data by performing operations on these matrices and extracting relevant information. The performance and cost objectives for the EO task can be represented by the formula 1:

$$\begin{aligned} t &= \tau(s, n, r) \\ p &= \upsilon(t, n, r) \\ \text{where } r &\in R;\ n, s \in N. \end{aligned} \tag{1}$$

where τ and υ are the performance and cost objective functions, respectively. t is the execution time of the EO task with complexity s regarding the input data size. The computational Dask cluster consists of n number of nodes, each with an instance type of r from the finite set of cloud-based computing instances R. s and n are positive integers from the set of natural numbers N. The cost of the EO task is represented by p. It depends on the execution time of the task and the characteristics of the computational cluster, particularly the number of nodes and instance type of the nodes. Formula 2 is used to find the optimal combination of the computational resources to balance performance and cost.

$$\begin{aligned} &\min_{r \in R} && \big[t = \tau(s, n, r),\, p = \upsilon(t, n, r)\big] \\ &\text{subject to} && 0 < t \leq t', 0 < p \leq p'. \end{aligned} \tag{2}$$

where t' and p' are corresponding task execution time and cost budget constraints.

3 Related work

EO data processing is a crucial aspect of many environmental research studies, which rely on vast amounts of satellite data to monitor resources such as land and water.

Several EO applications are highlighted in studies shoreline delineation services (Astsatryan et al., 2022) and regional crop classification (Ijabs and Urtāns, 2022). As the volume and complexity of the data have grown, the need for efficiency in EO data processing has become increasingly important. Hence, the limitation of these kinds of studies is that they do not focus on the efficiency of processing EO data, whereas handling large datasets requires significant computational resources and processing time. Many research studies have adopted scalable data processing frameworks and techniques to enhance data processing performance on the top of various distributed platforms and tools such as Spark, Dask, and Message Passing Interface (MPI) (Huang et al., 2017; Tan et al., 2021; Appel et al., 2018; Xu et al., 2020; Wang et al., 2020) . The studies indicate that scaling can enhance the performance of specific EO tasks without considering the optimal factor of scaling or configuration parameters provided by the platforms and tools.

To address this gap, researchers (Yu et al., 2021; Sun et al., 2019) have proposed methods to optimize EO task configuration parameters, search for an optimal distributed processing solution by considering factors such as the number of chunks and worker assignments through regression models and genetic algorithms. These studies show that optimizing configuration parameters can significantly decrease runtime compared to using the default parameters and find the optimal scaling factor considering limited computing resources. Nevertheless, the limitation of these studies is that they concentrate solely on the efficient performance objective without considering the expenses related to scaling. Therefore, a multi-objective optimization approach is necessary to balance performance and resource cost, considering factors such as execution time and cost depending on the complexity of the EO task.

4 Methodology

The multi-objective optimization service has been seamlessly incorporated into the EO data processing platform [1], as illustrated in Figure 1.

The following is the step-by-step workflow of the EO data processing platform:

1. The client submits a request to the processing engine with specific parameters such as the area of interest, time frame, desired function, optional parameters of execution time (t'), and cost (p') constraints;
2. The processing engine then forwards the request to the optimization service to determine the most efficient configurations of the Dask cluster;
3. The processing engine uses the optimization service's recommended configurations to establish a Dask cluster in the cloud through the Dask-gateway proxy. The configuration providing the best performance is selected if there are multiple optimal configurations considering performance and cost objectives;
4. The engine generates a computational graph considering the client's input and executes it in the Dask cluster. The Dask cluster begins processing the data, retrieving the required data from the EO repositories, and performing the computation in a distributed manner;

[1] https://github.com/ArmHPC/EO_data_multiobjective_Optimization

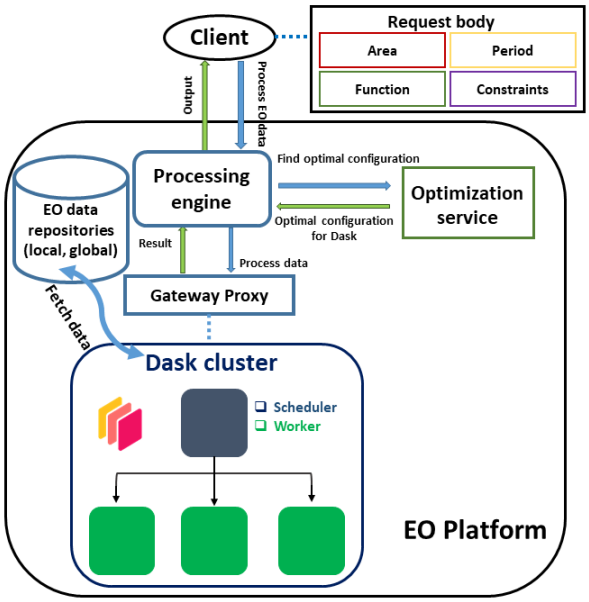

Fig. 1. The structure of the EO data processing platform.

5. Finally, the processed output is delivered to the client.

Dask-gateway [2] ensures the optimal setup recommended by the optimization service. It allows the possibility to create and handle Dask clusters, allowing for easy and flexible management of them remotely. Dask-gateway can be deployed on any cloud resource using cluster backends such as Kubernetes, Hadoop/YARN, and HPC job queues. We follow specific standards of data fetching to maintain compatibility with other services. We use the SpatioTemporal Asset Catalogs API (STAC-API) (Zhao et al., 2021) to access EO data from repositories, as it offers innovative solutions for searching, calculating queried data size, and generating a calculation graph for EO data processing using lightweight metadata in JSON format instead of heavy satellite image data. The Armenian data cube is used as a local EO data repository, providing researchers access to high-resolution satellite imagery from Landsat and Sentinel missions, covering the territory of Armenia (Asmaryan et al., 2019). This platform has been used for various experiments and studies to monitor and manage the region's natural resources (Astsatryan et al., 2021, 2022). As a global EO data repository, the Amazon Stack EO Data has been used as a cloud-based platform that provides access to vast EO data from various sources (Giuliani et al., 2017).

The optimization method leverages this capability to estimate the input data size using the parameters of interest and the time frame provided by the client. The work of the optimization service is shown in algorithm 1.

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

Algorithm 1 Optimization algorithm

Require: s, t', p' ▷ Task complexity, execution time and cost constraints
Ensure: $\min_{r \in R} \left[t = \tau(s, n, r), p = \upsilon(t, n, r) \right]$ subject to: $t \leq t', p \leq p'$
  $configs \leftarrow$ **finite set of Dask cluster configurations**
  $results \leftarrow \{\}$
  $optimalPoints \leftarrow \{\}$
  for $config$ **in** $configs$ **do**
    $time \leftarrow \tau(s, n, r)$ ▷ find or predict execution time for the given $config$ and complexity s
    $cost \leftarrow time \times config.instanceRate \times config.nodes$
    if $cost \leq p'$ **AND** $time \leq t'$ **then**
      $results$.append($(config, cost, time)$)
    end if
  end for
  for r **in** $results$ **do**
    $nonDominatedPoints \leftarrow \{r.cost > it.cost$ **AND** $r.time > it.time$ **for it in** $results\}$
    if $nonDominatedPoints$ **is empty then**
      $optimalPoints$.append(r)
    end if
  end for
  return $optimalPoints$

The optimization algorithm begins by evaluating the task execution time and the cost of the required computational resources (see formula 1) considering different Dask cluster configurations from a finite set (with n number of nodes and r instance type) and the calculated complexity of the task same as the input data size s. The process of assessing the execution time of the given task considering τ performance objective function involves examining the historical simulation dataset to verify whether similar simulations have been conducted with comparable complexity considering the input data size. If such data is absent from the historical simulation dataset, a pre-trained regression model predicts the execution time. Then the optimization service calculates the cost of each configuration using the objective function υ by multiplying the estimated execution time by the number of worker nodes and the hourly rate of the worker instance type. Suppose the resulting cost and execution time align with the client's restrictions (t' - execution time constraint and p' - cost constraint). In that case, it is added to a list of solution that satisfies the client's condition. The received list of configurations is filtered by eliminating items that are outperformed by others concerning cost and performance objectives. Specifically, configurations with higher cost and execution time are removed from the list. Once this operation is complete, the resulting list of non-dominated configurations comprises those that offer comparable cost and performance and do not have a significant advantage.

5 Evaluation

This section evaluates the suggested service and discusses the experiments' outcomes. The subsection "Experimental infrastructure" details the infrastructure used for simu-

lations. The ”Data and study area” subsection presents the satellite data and the area of interest utilized for the study. The ”Experimental settings” subsection presents the EO processing functions and input data characteristics. Finally, the ”Experimental results and discussion” subsection introduces and discusses the results obtained from the proposed optimization service.

5.1 Experimental infrastructure

The experiments made use of computational resources from both CloudLab (Duplyakin et al., 2019) and the Armenian cloud infrastructure (Astsatryan et al., 2015), which offer a diverse range of services to the communities (Astsatryan et al., 2016, 2013). CloudLab provided access to nearly 1,000 machines across three sites in the United States, while the Armenian cloud infrastructure offered more than 600 vCPUs. The proposed optimization service underwent a rigorous evaluation on multiple Dask clusters, encompassing a diverse array of worker nodes and instances (refer to table 1). These instances ranged from a single CPU with 2GB RAM up to 64 CPUs with 128GB RAM, and the same Dask configuration was applied to all possible instance types.

Combining Dask with Kubernetes provides a highly scalable and flexible computing infrastructure on top of the Cloudlab and Armenian cloud platforms. Kubernetes serves as a resource manager, dynamically allocating and releasing resources based on the requirements of Dask workflows. Dask Gateway offers a user-friendly web interface for managing Dask clusters, allowing users to specify the number of worker nodes, CPUs, and memory required. Users can connect to the Dask Gateway server using a configured Jupyter notebook. A Kubernetes pod with specified CPU and RAM requirements represents each worker node in the Dask cluster. All experimental results are stored in a historical simulation dataset. In an out-of-memory error, the execution time is recorded as infinite. Dask automatically partitions input data into chunks for more efficient processing, using a default chunk size of 128 MB with auto chunk sizing.

5.2 Data and study area

The study area, covering Armenia’s territory, was analyzed using the open-source Sentinel-2 satellite to obtain EO data. The satellite captures 12 different band images with varying wavelengths, providing rich information about the Earth’s surface. For this study, the Near-infrared, Red, Blue, and Green bands, labeled as NIR, RED, $BLUE$, and $GREEN$, were used as inputs for the EO data processing functions. We utilize the Sentinel-2 Cloud-Optimized GeoTIFFs repository[3] that provides access to the data via STAC API for efficient and scalable retrieval. This approach made it easy to access the data required for executing our EO data processing functions seamlessly.

Overall, using the Sentinel-2 satellite and STAC-API enable us to obtain high-quality EO data for our analysis, providing a solid foundation for the subsequent processing and analysis.

[3] https://registry.opendata.aws/sentinel-2-l2a-cogs/

5.3 Experimental settings

Time-series analysis is critical to EO monitoring, providing valuable benefits such as detecting water or land area changes over time. We selected several EO indices as data processing functions to evaluate the proposed optimization method, including the Normalized Difference Vegetation Index (NDVI) (Pettorelli et al., 2005), Normalized Difference Water Index (NDWI) (McFEETERS, 1996), and Enhanced Vegetation Index (EVI) (Huete et al., 1997) as specified in formula 3. These indices offer critical information about the condition and health of vegetation and can detect the presence of water bodies.

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

$$EVI = 2.5 \times \frac{NIR - RED}{NIR + 6 \times RED–7.5 \times BLUE + 1} \tag{3}$$

$$NDWI = \frac{GREEN - NIR}{GREEN + NIR}$$

The calculations required for these functions are matrix operations easily distributed across the cluster nodes, making them ideal for use in a distributed computing environment. This approach allows for the rapid and efficient processing of extensive data, providing the necessary insights for effectively monitoring the study area. To evaluate the optimization service, we consider three workloads with varying input data sizes, each corresponding to a different period for the Armenian territory. The selected workloads have weekly (light), monthly (medium), and seasonal (heavy) workloads. The characteristics of each workload are provided in Table 2.

Table 2. Workload characteristics

Workload type	Period	Size (Tb)
Light	Weekly	0.08
Medium	Monthly	0.32
Heavy	Seasonal	1.20

The weekly workload indicates a shorter period and requires less processing capacity. The medium-sized input data for the monthly workload corresponds to a more extended period and calls for a higher processing capacity than the weekly workload. Among the three workloads, the seasonal workload has the highest input data amount requiring the most processing power.

As part of the experiments, we compute the average NDVI, NDWI, and EVI values for the Armenian territory at weekly, monthly, and seasonal intervals. By considering various Dask cluster configurations, the study can assess how different processing setups affect the efficiency of the analysis.

5.4 Experimental results and discussion

The use of Dask clusters with varying numbers of nodes and instance types results in different processing times for the EO data processing workflow. Figure 2 shows the execution time of the average NDVI function and the resource cost for various Dask cluster configurations under the seasonal workload with varying execution times and costs.

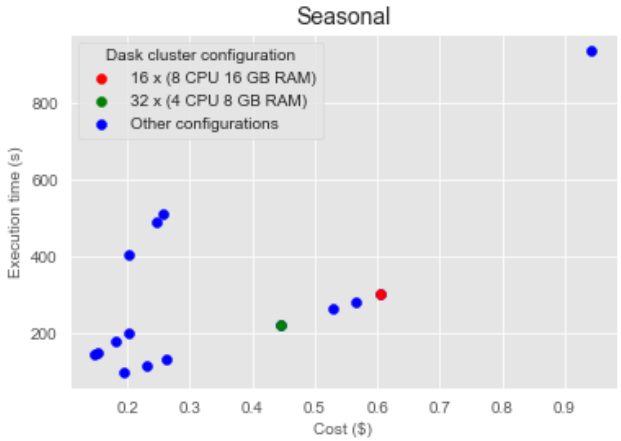

Fig. 2. Performance-cost relationship of different Dask configurations for a seasonal workload.

The experiments demonstrate that the same number of CPUs and RAM for two Dask clusters does not necessarily mean their performance will be similar. For example, the red point in the figure corresponds to 16 nodes with the 2xlarge instance (8 CPUs per node), while the green point corresponds to 32 nodes with the xlarge instance (4 CPUs per node) each, totaling 128 CPUs. However, their execution times and costs differ significantly. The green point demonstrates better execution time and lower cost than the red point. Thus, the green point is the better option, and it dominates the red one considering the idea of Pareto. The blue-colored configurations labeled as ”Other configurations” denote Dask clusters that include a variety of worker nodes across the selected instance types. Figure 3 demonstrates the Pareto front for the seasonal workload specifying a trade-off between competing performance and cost objectives.

To summarize the results of the optimization algorithm, it was found that only a tiny percentage of the configurations were considered Pareto-optimal and non-dominated for each workload. Specifically, for the seasonal workload, only 5.7% of the configurations were Pareto-optimal, with two optimal points identified - one with the best execution time and the other with the lowest cost. The number of Pareto-optimal solutions depends on the complexity of the studied Dask configurations. In contrast, only three Pareto-optimal configurations were identified from the available options for the weekly and monthly workloads. These evaluations demonstrate that the optimization

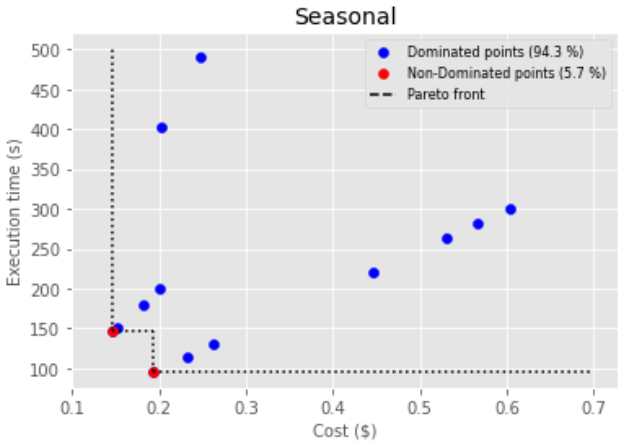

Fig. 3. Pareto front for the seasonal workload.

algorithm effectively identifies the best Dask cluster configurations for each workload. It can help users decide which configuration to choose based on their specific performance and cost objectives. Additionally, selecting from the optimal configuration can improve execution performance while reducing the cost of the computational resources. When examining the seasonal workload, the average execution time and cost for configurations with 128 CPU and 256 GB of RAM are 201 seconds and 0.405 $, respectively. In contrast, the optimal configurations provide 121 seconds average execution times and 0.17 $ cost. The comparison demonstrates how choosing an optimal point can increase performance by almost 1.66 times while cutting costs by a factor of 2.38 on average. The Pareto fronts for the weekly and monthly workloads are shown in figures 4 and 5, respectively.

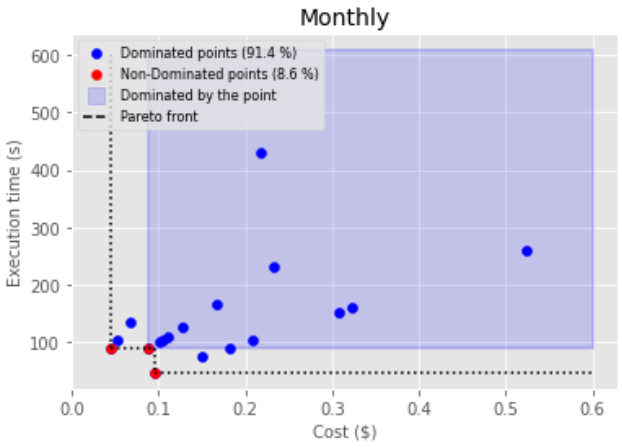

Fig. 4. Pareto-optimal points for the monthly workload.

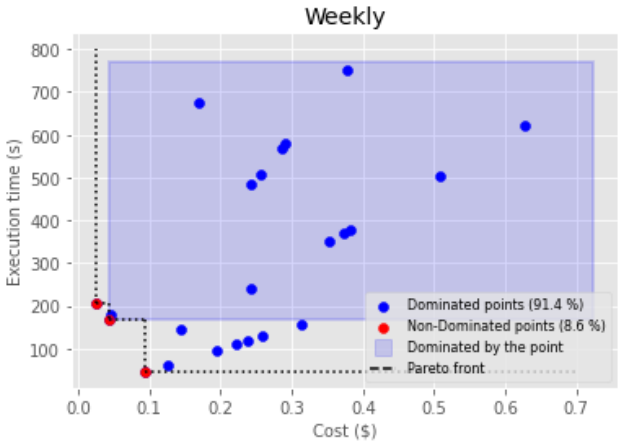

Fig. 5. Pareto-optimal points for the weekly workload.

The figure showcases the extent of a configuration's domination and the Pareto front. 91.4% are considered as dominated as the remaining 8.6% Pareto-optimal points outperform them. The optimization algorithm also considers execution time and cost constraints. Figure 6 displays the unique, non-dominated solution for the specified input restrictions with a cost of 0.3 $ and a performance of 150 seconds.

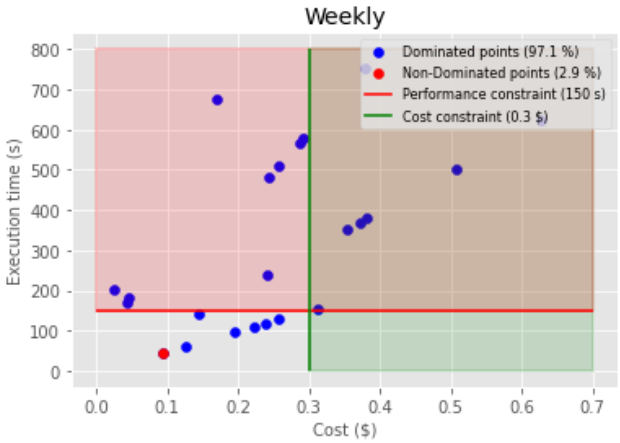

Fig. 6. Pareto-optimal point for the weekly workload considering execution time and cost constraints.

The figure identifies only one point as Pareto-optimal based on the given constraints. The results obtained from calculating EO functions are noteworthy, as the different functions showed similar execution speeds for the same input data. However, the input data size influenced the time taken to compute the functions, as depicted in Figure 7.

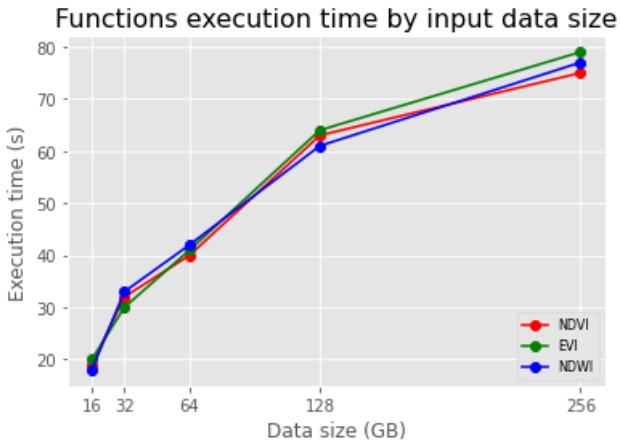

Fig. 7. EO functions execution time for the different input data sizes.

The figure illustrates the function execution time on an 8-node Dask cluster that employs the 2xlarge instance type. The time required for CPU calculations is comparatively negligible. Furthermore, the simulation results reveal a significant correlation between the processing time of EO data and the input data size, which follows an almost linear pattern. This suggests that a linear regression model can predict the execution time for an input data size that is not included in the historical dataset. This prediction can be integrated into a multi-objective optimization method.

The multi-objective optimization method relies on considering a finite range of Dask configurations that impact task execution times to function effectively with diverse input data sizes. To this end, experimental simulations are conducted with a selected set of finite Dask clusters, EO functions, and input data sizes. The results are recorded in a simulation dataset to train a regression model for predicting task execution time. The k-Fold Cross-validation method is used for hyperparameter tuning to optimize the model's performance. Table 3 presents some of the outcomes of the trained model tested on previously unseen data.

Table 3. Several prediction examples

Function	Data size (Tb)	Nodes	Instance type	Actual		Predicted		Accuracy
				Time (s)	Cost ($)	Time (s)	Cost ($)	
NDVI	0.05	2	large	431	0.027	349	0.022	81
	0.1	1	xlarge	1680	0.106	2016	0.127	80
EVI	0.2	4	2xlarge	88	0.044	77	0.039	85
	0.4	8	4xlarge	71	0.143	82	0.165	83
NDWI	0.6	4	8xlarge	213	0.429	175	0.353	82
	0.8	2	16xlarge	195	0.393	230	0.464	82

6 Conclusion

The multi-objective optimization service offers a promising solution for optimizing the configuration of Dask clusters for efficient EO data processing. The experiments demonstrate that the proposed method outperforms the traditional approaches and is a more efficient and cost-effective solution. The findings indicate that many configurations may not be effective when considering performance and cost objectives. However, choosing one of the optimal configurations recommended by the optimization service can result in a substantial improvement in performance and a reduction in cost, particularly in the case of the seasonal workload, where there can be a performance improvement of up to 1.66 times and a cost reduction of 2.38 times. This highlights the importance of the multi-objective optimization approach for EO data processing workflows.

Acknowledgements

The research was supported by the Science Committee of the Republic of Armenia by the project entitled "Scalable data processing platform for EO data repositories" (Nr. 22AA-1B015) and the University of Geneva Leading House by the projects entitled "Self-organized Swarm of UAVs Smart Cloud Platform Equipped with Multi-agent Algorithms and Systems" (Nr. 21AG-1B052), "Remote sensing data processing methods using neural networks and deep learning to predict changes in weather phenomena" (Nr. 21SC-BRFFR-1B009), and "ADC4SD: Armenian Data Cube for Sustainable Development".

References

Appel, M., Lahn, F., Buytaert, W., Pebesma, E. (2018). Open and scalable analytics of large earth observation datasets: From scenes to multidimensional arrays using scidb and gdal, *ISPRS Journal of Photogrammetry and Remote Sensing* **138**, 47–56.

Asmaryan, S., Muradyan, V., Tepanosyan, G., Hovsepyan, A., Saghatelyan, A., Astsatryan, H., Grigoryan, H., Abrahamyan, R., Guigoz, Y., Giuliani, G. (2019). Paving the way towards an armenian data cube, *Data* **4**(3).

Astsatryan, H., Grigoryan, H., Abrahamyan, R., Poghosyan, A., Asmaryan, S., Muradyan, V., Tepanosyan, G., Guigoz, Y., Giuliani, G. (2022). Shoreline delineation service: using an earth observation data cube and sentinel 2 images for coastal monitoring, **15**, 1587–1596.

Astsatryan, H., Grigoryan, H., Poghosyan, A., Abrahamyan, R., Asmaryan, S., Muradyan, V., Tepanosyan, G., Guigoz, Y., Giuliani, G. (2021). Air temperature forecasting using artificial neural network for ararat valley, *Earth Science Informatics* **14**, 1–12.

Astsatryan, H., Lalayan, A., Giuliani, G. (2023). Scalable data processing platform for earth observation data repositories, *Scalable Computing: Practice and Experience* **24**(1), 35–44.

Astsatryan, H., Narsisian, W., Asmaryan, S. (2016). Swat hydrological model as a daas cloud service, *Earth Science Informatics* **9**, 401–407.

Astsatryan, H., Sahakyan, V., Shoukourian, Y., Dongarra, J., Cros, P.-H., Dayde, M., Oster, P. (2015). Strengthening compute and data intensive capacities of armenia, *2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER)*, pp. 28–33.

Astsatryan, H., Sahakyan, V., Shoukouryan, Y., Daydé, M., Hurault, A., Guivarch, R., Terzyan, H., Hovhannisyan, L. (2013). On the easy use of scientific computing services for large scale linear algebra and parallel decision making with the p-grade portal, *Journal of grid computing* **11**, 239–248.

Camps-Valls, G., Verrelst, J., Munoz-Mari, J., Laparra, V., Mateo-Jimenez, F., Gomez-Dans, J. (2016). A survey on gaussian processes for earth-observation data analysis: A comprehensive investigation, *IEEE Geoscience and Remote Sensing Magazine* **4**(2), 58–78.

Duplyakin, D., Ricci, R., Maricq, A., Wong, G., Duerig, J., Eide, E., Stoller, L., Hibler, M., Johnson, D., Webb, K., Akella, A., Wang, K., Ricart, G., Landweber, L., Elliott, C., Zink, M., Cecchet, E., Kar, S., Mishra, P. (2019). The design and operation of CloudLab, *Proceedings of the USENIX Annual Technical Conference (ATC)*, pp. 1–14.

Giuliani, G., Chatenoux, B., de Bono, A., Rodila, D., Richard, J.-P., Allenbach, K., Dao, H., Peduzzi, P. (2017). Building an earth observations data cube: lessons learned from the swiss data cube (sdc) on generating analysis ready data (ard), *Big Earth Data* **1**, 1–18.

Huang, W., Meng, L., Zhang, D., Zhang, W. (2017). In-memory parallel processing of massive remotely sensed data using an apache spark on hadoop yarn model, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **10**(1), 3–19.

Huete, A., Liu, H., Batchily, K., van Leeuwen, W. (1997). A comparison of vegetation indices over a global set of tm images for eos-modis, *Remote Sensing of Environment* **59**(3), 440–451.

Ijabs, H., Urtāns, Ē. (2022). Bidirectional long short-term memory networks for automatic crop classification at regional scale using tabular remote sensing time series, *Baltic Journal of Modern Computing* **10**(4), 611–622.

Li, Z. (2020). *Geospatial Big Data Handling with High Performance Computing: Current Approaches and Future Directions*, High Performance Computing for Geospatial Applications, pp. 53–76.

Malla, S., Christensen, K. (2019). Hpc in the cloud: Performance comparison of function as a service (faas) vs infrastructure as a service (iaas), *Internet Technology Letters* **3**, e137.

McFEETERS, S. K. (1996). The use of the normalized difference water index (ndwi) in the delineation of open water features, *International Journal of Remote Sensing* **17**(7), 1425–1432.

Pettorelli, N., Vik, J. O., Mysterud, A., Gaillard, J.-M., Tucker, C. J., Stenseth, N. C. (2005). Using the satellite-derived ndvi to assess ecological responses to environmental change, *Trends in Ecology and Evolution* **20**(9), 503–510.

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling, *Proceedings of the 14th python in science conference*, Vol. 130, SciPy Austin, TX, p. 136.

Sun, J., Zhang, Y., Wu, Z., Zhu, Y., Yin, X., Ding, Z., Wei, Z., Plaza, J., Plaza, A. (2019). An efficient and scalable framework for processing remotely sensed big data in cloud computing environments, *IEEE Transactions on Geoscience and Remote Sensing* **57**(7), 4294–4308.

Tan, X., Di, L., Zhong, Y., Yao, Y., Sun, Z., Ali, Y. (2021). Spark-based adaptive mapreduce data processing method for remote sensing imagery, *International Journal of Remote Sensing* **42**(1), 191–207.

Wang, J., Huang, X., Zheng, J., Rajapakshe, C., Kay, S., Kandoor, L., Maxwell, T., Zhang, Z. (2020). Scalable aggregation service for satellite remote sensing data, *in* Qiu, M. (ed.), *Algorithms and Architectures for Parallel Processing*, Springer International Publishing, Cham, pp. 184–199.

Xu, D., Ma, Y., Yan, J., Liu, P., Chen, L. (2020). Spatial-feature data cube for spatiotemporal remote sensing data processing and analysis, *Computing* **102**.

Yao, X., Li, G., Xia, J., Ben, J., Cao, Q., Long, Z., Yue, M., Zhang, L., Zhu, D. (2019). Enabling the big earth observation data via cloud computing and dggs: Opportunities and challenges, *Remote Sensing* **12**, 62.

Yu, Z., Wang, Z., Bai, L., Chen, L., Tao, J. (2021). Parameter optimization on spark for particulate matter estimation, *2021 Workshop on Algorithm and Big Data*, Association for Computing Machinery, United States, pp. 9–13.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S., Stoica, I. (2016). Apache spark: A unified engine for big data processing, *Commun. ACM* **59**(11), 56–65.

Zhao, Y., Yang, X., Vatsavai, R. R. (2021). A scalable system for searching large-scale multi-sensor remote sensing image collections, pp. 3780–3783.

Received February 25, 2023 , revised June 6, 2023, accepted August 7, 2023