CLOUD-BASED MODULAR SYSTEM FOR ACQUISITION AND VISUALIZATION OF OCO-2 REMOTELY SENSED CO_2 DATA

Maroš Valášek[1,*], Roman Budjač[2], Martin Hanzely[3], Neven Vrček[4]

[1]Department of Control and Information Systems, Faculty of Electrical Engineering and Information Technology, University of Zilina, Zilina, Slovakia
[2]Research Centre, University of Zilina, Zilina, Slovakia
[3]Travelco s.r.o., Oscadnica, Slovakia
[4]Department of Information Systems Development, University Zagreb, Varazdin, Croatia

*E-mail of corresponding author: maros.valasek@feit.uniza.sk

Maros Valasek 0009-0006-4781-4628,
Martin Hanzely 0009-0002-5383-7288,
Roman Budjac 0000-0001-6840-1706,
Neven Vrcek 0000-0002-4037-1522

Resume

In this paper is presented a cloud-based framework for automated acquisition and visualization of Orbiting Carbon Observatory-2 (OCO-2) satellite CO_2 data. The system employs an ETL pipeline with OPeNDAP protocol for selective data retrieval, reducing network overhead while processing L2 Standard and L2 Lite FP products. Built on Amazon Web Services (AWS) infrastructure, using Python (Pandas, Dash, Plotly) and Docker orchestration, the modular architecture implements dependency injection for runtime flexibility. The deployed system achieves daily automated ingestion with 2.25 km × 1.29 km spatial resolution, enabling the real-time monitoring through interactive web visualization. The system is designed as a foundation for future analytical research, providing ready integration points for machine learning models to perform advanced CO_2 pattern recognition and predictive analysis.

Article info

Received 14 November 2025
Accepted 5 December 2025
Online 16 January 2026

Keywords:

CO_2
data acquisition
remote sensing
machine learning

Available online: https://doi.org/10.26552/com.C.2026.013

ISSN 1335-4205 (print version)
ISSN 2585-7878 (online version)

1 Introduction

The current level of CO_2 in the atmosphere has increased in the last decades significantly, necessitating advanced monitoring techniques such as those offered by satellite-based remote sensing [1]. These platforms offer a unique vantage point for comprehensive atmospheric observations, enabling the precise quantification of greenhouse gas concentrations and their spatial and temporal variations [2]. This approach is critical for identifying emission hotspots, such as large urban centers, power plants, and industrial facilities, which are major contributors to anthropogenic CO_2 emissions [3-4]. However measurements are not consistent and not all localities have the same measurement fidelity, with current methods facing challenges in distinguishing anthropogenic CO_2 signals from background concentrations and limitations in spatio-temporal resolution [5]. Furthermore, the distribution of CO_2 satellite measurements exhibits significant latitudinal bias [6]. This necessitates the integration of data from diverse observational platforms, including ground-based in-situ sensors and atmospheric models, to enhance the accuracy and robustness of CO_2 emission estimates [7]. Most high-quality observations are concentrated in the Northern Hemisphere, especially between 30°N and 70°N, where the industrial activity is most intensive. In the future, a more uniform global coverage is essential for a comprehensive understanding of carbon cycle dynamics and for supporting climate mitigation strategies [8]. This uneven coverage creates substantial data gaps, especially in polar regions and over the oceans, which play a crucial role in carbon sequestration. This fact limits our understanding of global carbon cycle dynamics. To address these limitations, a scalable cloud-based data storage and visualization tool is imperative for managing the multi-type geospatial data generated from satellite-derived CO_2 flux measurements [9].

Transportation represents one of the most significant sources of anthropogenic CO_2 emissions, contributing nearly a quarter of global energy-related greenhouse gases. Freight road activity alone accounts for a

substantial portion of these emissions, approximately 29.4% of all the transport-related carbon output [10]. Accurate monitoring of CO_2 in transport corridors, urban traffic zones, and major logistic hubs is therefore essential for assessing the environmental impact of mobility systems and evaluating the effectiveness of decarbonization policies. Leveraging modern cloud-based solutions can overcome the complex demands of air quality management and policy monitoring by providing advanced tools for environmental monitoring and policy formulation [11]. Satellite-based CO_2 measurements provide valuable large-scale insights into emission patterns from road, air, and maritime transport, complementing ground-based inventories that often lack spatial coverage or temporal consistency. Specifically, satellites like the Orbiting Carbon Observatory-2 offer high-resolution spatiotemporal data of CO_2 concentrations [12].

Global energy consumption is strongly coupled with CO_2 emissions: as the number of IoT devices, deployed for CO_2 monitoring, continues to increase, the associated energy demand likewise increases and is still largely met by fossil-based generation within the current energy mix. These trends motivate the use of hybrid energy-harvesting techniques to enable large-scale sensor networks to operate more sustainably, without further amplifying the carbon footprint of the monitoring infrastructure [13].

In this paper, the development of a cloud-based system, specifically designed for the acquisition and transmission of carbon dioxide data from satellite sources to a custom visualization platform, is detailed. This system leverages Amazon Web Services for its scalable and flexible infrastructure, enabling the robust data processing and monitoring capabilities [14]. The architecture integrates satellite-derived CO_2 measurements, facilitating their secure streaming and storage within the AWS ecosystem [15].

2 Related works

The cloud-based Earth Observation (EO) data processing platforms have evolved significantly to address the challenges of managing and analyzing large-scale satellite datasets. The openEO initiative represents a key advancement in this field, establishing an open API standard that abstracts away infrastructure complexities. Developed within the H2020 project (2017-2020) [16], openEO provides a unified framework for accessing diverse satellite data sources through high-level abstractions that treat image collections as data cubes, enabling scientists to focus on analysis rather than data handling. The Application Programming Interface (API) supports multiple client libraries (Python, R, JavaScript, QGIS) and integrates with existing image analysis services.

Recent studies highlight the growing need for structured data processing architectures to manage the massive volumes of geospatial and satellite information generated by modern EO missions. Romero et al. [17] introduced an ETL-based framework for integrating remote sensing data from multiple satellites, demonstrating a modular architecture that enables extraction, transformation, and loading of heterogeneous data formats (e.g., NetCDF, HDF, GeoTIFF) into unified analytical repositories. Implemented in Python, and compatible with workflow systems, such as Apache Airflow and Dagster, their design facilitates flexible and scalable data fusion across instruments like GOES-16 and CloudSat, though its focus remains primarily on cloud profiling and radiometric data.

Similarly, Boudriki Semlali and El Amrani [18] proposed a hybrid ETL and stream-processing architecture combining the SAT-ETL-Integrator for satellite data preprocessing with the SAT-CEP-Monitor for real-time analysis based on Complex Event Processing (CEP). Their system integrates multisource datasets from NASA, NOAA, and ESA satellites with ground-based observations, performing multi-stage ETL operations and CEP aggregation to compute air-quality indices in near-real time. The approach demonstrated high processing efficiency and strong agreement with ground-based data ($r = 0.75$), underscoring the potential of ETL + CEP frameworks for environmental satellite data analytics.

While these works advance data integration and real-time processing, existing architectures remain limited in handling satellite-based CO_2 column measurements. For example, Boussaada et al. [19] describe a multilayer system architecture comprising *Data Acquisition*, *Data Processing and Communication*, *Data Storage*, and *Application Layer*, that primarily targets ground-based IoT sensors and citizen monitoring.

In contrast, this new approach extends the architectural model with dedicated modules for satellite CO_2 data, including an ETL pipeline, georeferencing, spatio-temporal interpolation, and integration with emission and meteorological models. This design establishes a systematic preprocessing layer for satellite-derived CO_2 datasets, addressing a critical gap in current EO data processing frameworks.

3 Materials and methods

In this section, an innovative and methodological framework for the systematic acquisition, processing, and interactive spreading of the large-scale atmospheric carbon dioxide (CO_2) data was introduced. Conventional approaches often face challenges related to voluminous data transfer, inflexible processing pipelines, and the presentation of findings in static, non-interactive formats. The system presented in this work addresses these limitations by combining a selective data-access mechanism with a modular, high-performance processing

pipeline and a dynamic, web-based visualization interface. The research led up to the instantiation of this framework as a web application for visualizing global CO_2 concentrations derived from satellite measurements, thereby demonstrating its efficacy and utility.

The main functions of the data module are the acquisition, processing, analysis, and storage of data from satellite measuring stations. One of the first tasks, therefore, was to find a suitable data source and secure programmatic access to the storage. We designated the Orbiting Carbon Observatory-2 (OCO-2) satellite measuring station as the primary data source. The measured data is stored in a publicly accessible repository managed by the Goddard Earth Sciences Data and Information Services Center (GES DISC) at NASA Goddard Space Flight Center. The data available in the storage facility are at various levels of processing, typically 0 to 4. The proposed application works with level 2 data, which are mapped to the relevant latitude and altitude and contain error indicators that specify whether the data are suitable for use. Two types of data products are the most suitable for processing: L2 Standard and L2 Lite FP. The L2 Standard data product is usually available one to two days after the measurement itself, but only for the current calendar year. The L2 Lite FP represents condensed data files that have undergone more thorough correction. Those files have been available approximately one month after the measurement itself since 2014. In this case, one data file represents one day, which allows for easier processing compared to L2 Standard files. In the proposed application, L2 Standard data was selected for daily downloads of new data and L2 Lite FP for downloading historical data.

There are several ways to access data. For the purposes of the proposed application, the OPeNDAP (Open-source Project for a Network Data Access Protocol) protocol was chosen. This is an open protocol for accessing data files based on http technology [20]. This protocol allows only selected variables of data files to be downloaded in NetCDF format, which makes it possible to work with a much smaller volume of data, resulting in less network load and faster processing. On the data provider side, the OPeNDAP protocol uses Hyrax server software with support for THREDDS catalogs [21]. The THREDDS catalogs are XML files that organize and describe data files and folders in the repository, making it possible to navigate the repository folders and find the desired files by date [22]. The storage has a root folder with a fixed address. The root folder, like all others, contains a catalog with a list of files, their address for access via the OPeNDAP protocol, and a list of nested folders. Access to data files depends on the selected type of data product, L2 Standard or L2 Lite FP. The folder structure and file naming are different in both cases. The program must be designed to work with both cases. The application must also include a function that works with these catalogs. The following flowchart shows the process of obtaining the addresses of the required data files from the THREDDS catalog.

The Requests library is employed to access the data repository, as it provides essential authentication support and significantly simplifies interface for working with HTTP protocol compared to Python's standard library. The retrieved data files must subsequently be converted to DataFrame objects implemented by the Pandas library. A DataFrame is a highly efficient data structure specifically designed for tabular data, rigorously supports critical operations such as filtering, such as filtering, aggregations and statistical operations. Pandas itself is a high-level data manipulation library, primarily engineered for advanced data analysis tasks. Its core components are implemented in the C programming language, which ensures high performance, while maintaining the simplicity and flexibility of Python.

After the processing and analysis, the obtained and transformed data need to be stored in a persistent storage system. Several storage options were designed for the needs of the application, such as CSV file stored on a local disk or Parquet file stored on a network-accessible data repository.

The first option, a CSV file on the local disk, is primarily intended for local testing, as CSV files are text based and human readable without requiring specialized software. In contrast, the Parquet files are binary and not directly readable without required tools, however, they provide higher efficiency for reading and writing operations and require less disk space. They are, therefore, more suitable for cloud storage environments.

For the data processing, the Extract, Transform, Load (ETL) process was adopted. This process consists of three main phases: extraction, transformation, and loading of the data.

The extraction phase involves acquiring data from one or more internal or external sources that may contain structured or unstructured data. It includes the initial validation of received data, ensuring that they are in the correct format and structure. During the transformation phase, various functions and rule sets are applied to the data, including filtering, filling in missing values, aggregation, and merging. The loading phase refers to storing the processed data in internal storage, which may include a file system, database, or data warehouse. All acquired data are stored to enable subsequent analysis.

Within this process, a function for analyzing the newly acquired and cleaned data is also executed. During this analysis phase, various aggregations and computations of supplementary information are performed, operations that would be inefficient to execute during each visualization. The analyzed data are likewise stored in the internal storage for further use. In addition, the processed data are continuously supplied to external machine learning model using the provider interface for further analysis and predictions.

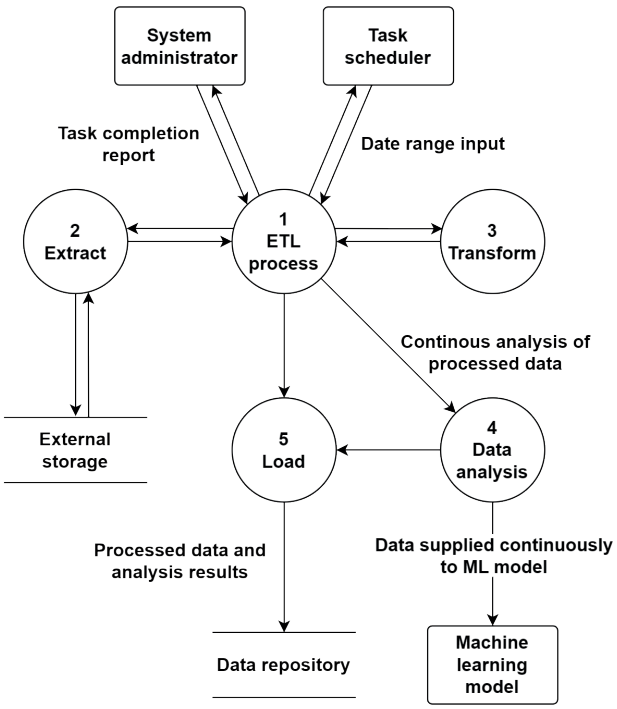

Figure 1 *Data Flow Architecture of the ETL Pipeline with integrated Data Analysis Module*

Figure 1 illustrates the main components and interactions within the ETL workflow. The central ETL process receives user-defined inputs through the task scheduler and system administrator, which trigger extraction, transformation, and loading operations. The Extract module retrieves raw data from external storage, while the Transform module performs preprocessing and normalization of the extracted datasets. The processed data is then passed to the Data analysis component, which is connected to the machine learning model and subsequently stored in the database through the *Load module*. Feedback on the task completion is returned to the system administrator, ensuring the controlled execution and monitoring of the ETL pipeline.

Figure 2 illustrates the class diagram of the application's data module. The entire ETL process is managed by the ETL Pipeline class through the dependencies *extract_strategy* and *load_strategy*, which are provided to the ETL Pipeline object via its constructor, depending on the configuration of the runtime environment. The data module was designed in accordance with the principles of object-oriented programming: abstraction, encapsulation, polymorphism, and inheritance.

The abstract class *BaseExtracto*r provides a basic interface for the data acquisition. The method *extract_date_range* accepts a date range as its parameter specifying the period for which data should be downloaded, and returns a pair consisting of the date and a DataFrame object. The actual implementation of data retrieval is encapsulated within concrete subclasses.

The TestExtractor class represents the straightforward implementation of the abstract BaseExtractor class, operating without access to real online data. It returns a *DataFrame* object supplied through the constructor and is used exclusively for automated testing purposes. This allows testing of the ETL Pipeline logic with controlled input data and without the need for an internet connection.

The abstract class *BaseOpendapExtractor* overrides the constructor of *BaseExtractor* by introducing a mandatory parameter, *opendap_client*, of type *OpendapClient*. The *OpendapClient* class defines and implements a simple interface for communication with a server via the OPeNDAP protocol and for interaction with the THREDDS catalog.

The classes *OpendapExtractor_L2Standard* and *OpendapExtractor_L2LiteFP* implement access to the actual data repository, managing authentication against the storage system and searching for files based on the specified date range.

The abstract class *BaseLoader* provides the fundamental interface for the data storage. The method save_dataframe accepts a parameter of type *DataFrame* and a desired filename, which it saves to persistent storage. Conversely, the method retrieve_dataframe accepts the name of the file to be opened and returns a *DataFrame* object. The retrieve_dataframe method is also utilized in the design of the application's visualization module. The implementation of the data storage operations is encapsulated within individual concrete subclasses.

The *TestLoader* class represents the implementation of the abstract *BaseLoader* class, operating without the use of a database system or file storage. It works exclusively with a *DataFrame* object in system memory,

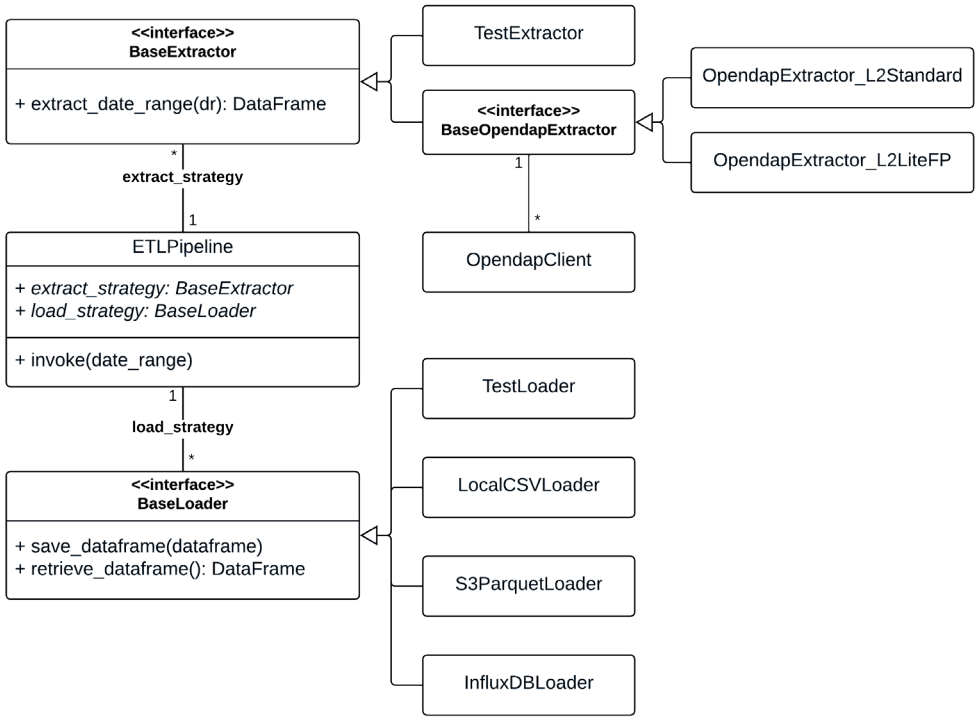

Figure 2 *Class diagram of the data module for application*

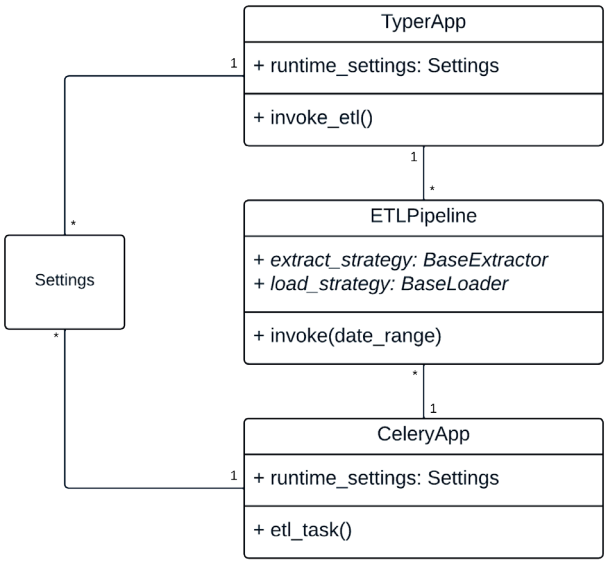

Figure 3 *Class diagram of data module for application with ETL Pipeline*

allowing testing of the ETLPipeline logic without the overhead of time-consuming data storage operations.

The *LocalCSVLoader* class stores the provides *DataFrame* object as a CSV file on disk and retrieves data only from that file. It is primarily intended for local testing of the application.

The *S3ParquetLoader* class stores the provides *DataFrame* object as a Parquet file in a data repository. The design assumes the use of Amazon Web Services (AWS) S3 storage, as the application is intended to be deployed in the AWS environment. This storage system supports file versioning. When a file is overwritten, its previous version is preserved and can be restored in cases such as data corruption. Therefore, the implementation of *S3ParquetLoader* also requires an interface for accessing the S3 storage.

The ETL Pipeline class itself provides a minimal interface, exposing only the invoke function, which initiates the process and manages the provided concrete dependencies. The design follows the Dependency Injection pattern, in which an object receives its dependencies externally rather than creating them

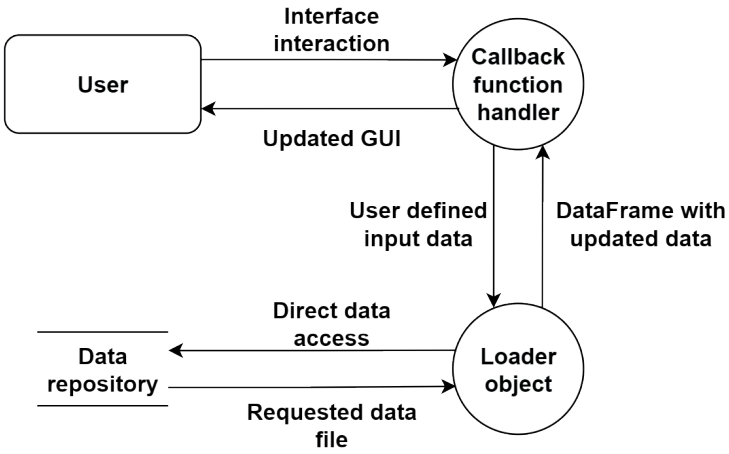

Figure 4 *Class diagram of data module for application with ETL Pipeline*

internally. Injecting dependencies that implement a common interface into the controller class allows the replacement of components at runtime without modifying the controller's source code. The ETL process in the application can be initiated from two sources: the Command-Line Interface (CLI) or a task scheduler. The CLI interface is managed by the Typer library, which simplifies the definition and parsing of command parameters. This allows for selection of a specific subclass of type BaseExtractor at the moment the command is executed. The function called by the command invoke-etl is defined in Figure 3. The pipeline_factory function supplements the required dependencies and returns an ETLPipeline object ready for execution. The example code below illustrates the *invoke_etl* function, which is triggered by the invoke-etl command within the CLI environment. Besides specifying the input dates, the user can also select the class used for data extraction.

Another method of triggering the ETL process is through a task scheduler. For this purpose, the Celery library is deployed, which enables the execution of scheduled and background tasks within the application. The configuration file defines the schedule read by Celery at startup. The schedule includes a single task, *daily_etl_task_L2_Standard*, which runs daily at 06:00, along with the definition of the task itself. The provided code snippet represents the Celery configuration, including the setup of periodic task scheduling.

3.1 Visualization module design

The main function of the visualization module is to display the results of data analysis to the user in the form of a web interface. Consequently, it is necessary to select appropriate software libraries for implementing an HTTP server and for generating and rendering various types of plots. For this purpose, the Dash and Plotly libraries were chosen. Both are open-source software libraries developed by Plotly Technologies Inc. and are freely available. They provide a simple interface for working with graphical visualizations and offer seamless mutual integration.

An application built using the Dash library functions as a lightweight web interface that listens on a predefined port, processes the HTTP requests, and returns responses. The user interface (UI) is defined using Python code. The library includes an html module, whose classes are translated at runtime into the corresponding HTML elements. Objects of these html classes are inserted into the application's layout attribute along with their properties. Based on this declarative structure, the corresponding HTML user interface is generated automatically.

The library provides a variety of interactive components, such as buttons and forms, which can be utilized in the application's design. Interaction with these elements is linked to specific application functions through unique element identifiers. Such a function is referred to as a callback and governs the modification of one or more HTML elements in response to user interaction - for example, re-rendering graphs with updated data when the user changes the selected date range. The design of these callback functions should follow the functional programming paradigm, meaning that they should not modify the internal state of the application but instead return output based only on input parameters.

The visualization components of the user interface are rendered using the Plotly library, which supports a wide range of chart types, including bar charts, line charts, maps, and correlation diagrams. Plotly functions accept *DataFrame* objects as their primary data source, allowing direct integration with the *BaseLoader* objects introduced in the previous section for data access.

Figure 4 presents a conceptual data flow diagram, illustrating the user interaction with the graphic user interface and sequential data processing components. The callback function handler accepts the input and calls the loader object for updated data. Loader object plays an intermediary role between the callback handler and a data repository. Such architecture allows for

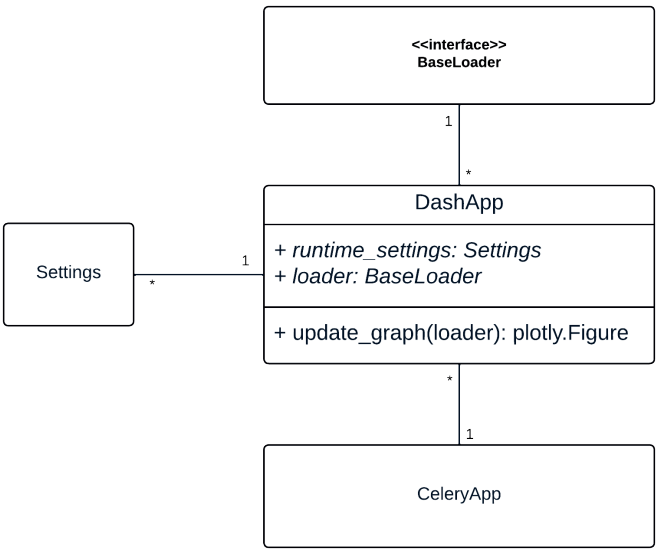

Figure 5 *Class diagram for the visualization module of application*

various persistent storage implementations. Loader then fulfills the query and passes the data to the callback handler, which then updates the graphic user interface accordingly.

A function that responds to user interaction must repeatedly establish connections to the database or the local file system and query large volumes of data. Consequently, situations may arise in which the response time exceeds the default timeout settings of the web server or the user's browser. Furthermore, a large number of long running calls could potentially overload all threads of the web application process. To mitigate these issues, background execution is employed.

Upon user interaction, the interactive input elements are temporarily disabled, and the execution request is placed into a dedicated task queue. Once the task is completed, the relevant element of the user interface is updated accordingly. The task queue management and sequential execution of commands are handled by the Celery library, which also serves as a task scheduler in the application's data module.

Figure 5 presents the class diagram of the application's visualization module. The abstract class *BaseLoader*, similar to its counterpart in the data module, provides an interface for loading data from a database or local disk. A specific implementation object is injected as a dependency into the *update_graph* function. This dependency injection design allows for the *update_graph* function logic to be tested without requiring an active database connection. As in the data module, a Settings class is also used and injected as a dependency during the application's startup.

Figure 6 displays a system architecture diagram with the technologies used to support the system. The application integrates external data sources from NASA GES DISC, including the OCO-2 L2 LITE FP and OCO-2 L2 STANDARD datasets. Those datasets are accessed through THREDDS Catalogue and OPENDAP NetCDF4 services.

The core of the architecture is Celery, a distributed task queue that operates as a worker to handle scheduled tasks for the data processing module. Data is stored in AWS S3 cloud storage and structured using Apache Parquet for optimal performance and accessibility. The Plotly App serves as the user interface, facilitating initial data preload and user interactions. Celery's callback functions are specifically used to respond to user interface requests, enabling seamless communication between the data processing module and the user interface. Redis is employed for caching and message brokering, enhancing system performance and responsiveness. Additionally, a machine learning service provider is integrated to supply the advanced analytics and enhance data analysis capabilities. This architecture emphasizes the seamless flow of data from external sources through processing and storage to user interaction, leveraging the robust open-source technologies to support environmental data processing and distribution.

4 Results

After the deployment, the application operated as expected. It downloaded the new data daily from the L2 Standard data product, provided that the data were available. If no data were accessible, the application reported an error, which was automatically communicated to the system administrator via the Sentry monitoring service. For historical data dating back to the beginning of 2023, the L2 Lite FP data product was utilized. Although those data were uploaded to storage with a monthly delay, they were of higher quality and underwent more rigorous validation procedures.

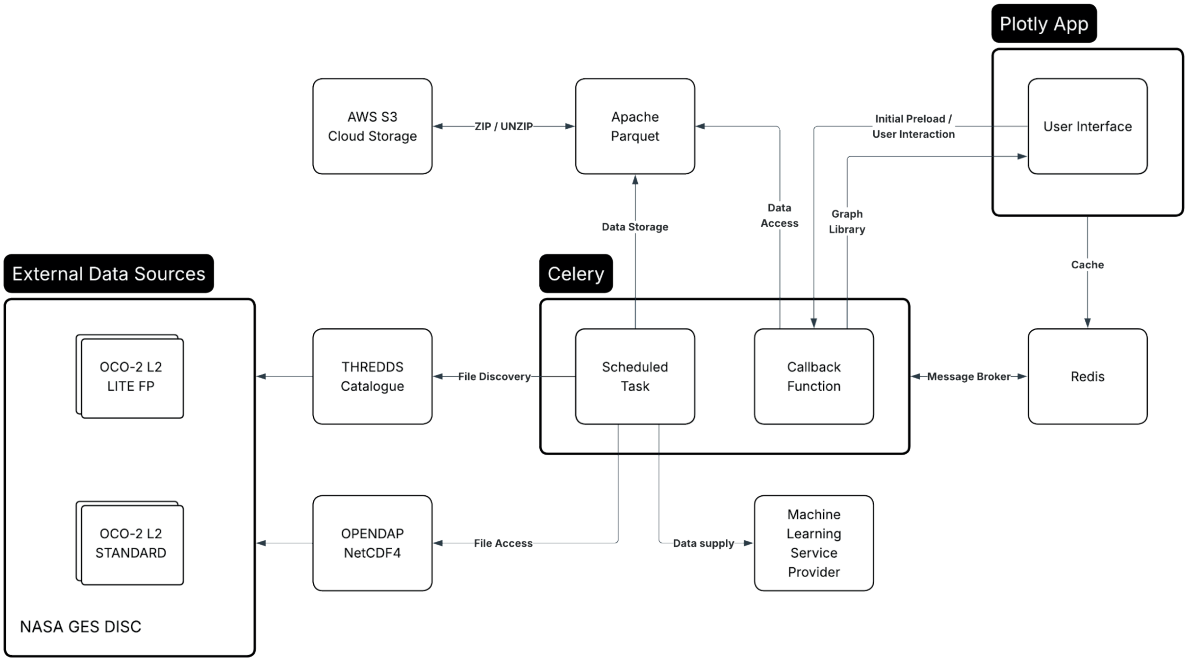

Figure 6 *The class diagram for GUI module of application*

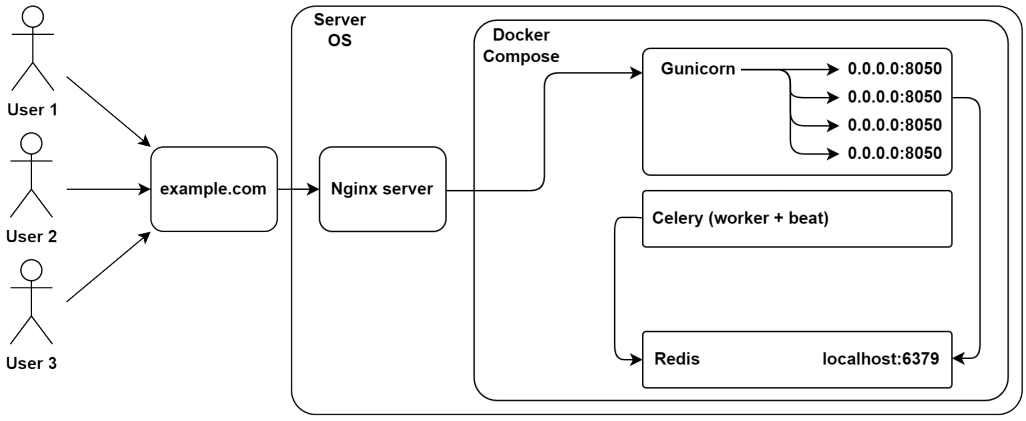

Figure 7 *System architecture of the container-based deployment environment*

The spatial resolution of the satellite sensors is 2.25 km × 1.29 km. Consequently, the acquired data can also be used for analyses of smaller geographic regions, provided that sufficient satellite coverage is available. A preliminary analysis of the dataset indicates that the satellite-based measurements closely follow the trends observed in ground-based monitoring stations, including seasonal variations and hemispheric extremes. Furthermore, the dataset enables the observation of global trends and anomalies, as well as the identification of sources, sink regions, and regional variations in atmospheric carbon dioxide concentration throughout the year.

The use of satellite data has also made it possible to isolate the CO_2 concentration values over the territory of Slovakia, a region where no open access ground-based measurement stations are available. These data can be employed for the more complex analyses of regional and global trends or for predictive modeling; however, such analyses are typically conducted in conjunction with meteorological and other complementary models.

Figure 7 illustrates the cloud-based deployment of the ETL and data processing services using Docker Compose. Incoming HTTP requests are routed through an Nginx reverse proxy running on the host operating system. The proxy forwards the requests to a Gunicorn application server that handles multiple worker instances on port 8050. The background data processing tasks are managed by a Celery service (worker and scheduler), which interacts with two supporting components: All the components run as Docker containers orchestrated within a single server environment, ensuring scalability, modularity, and fault isolation across services.

The examples in Figures 8 and 9 show the user interface of application for the two different user-

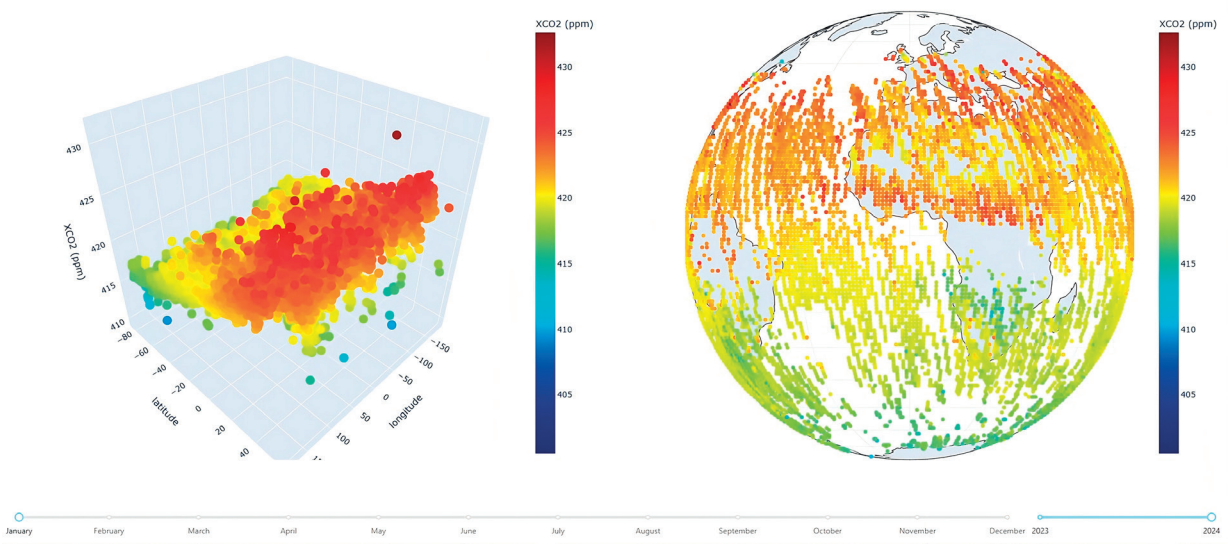

Figure 8 *Graph and visualization for average monthly values - January 2024*

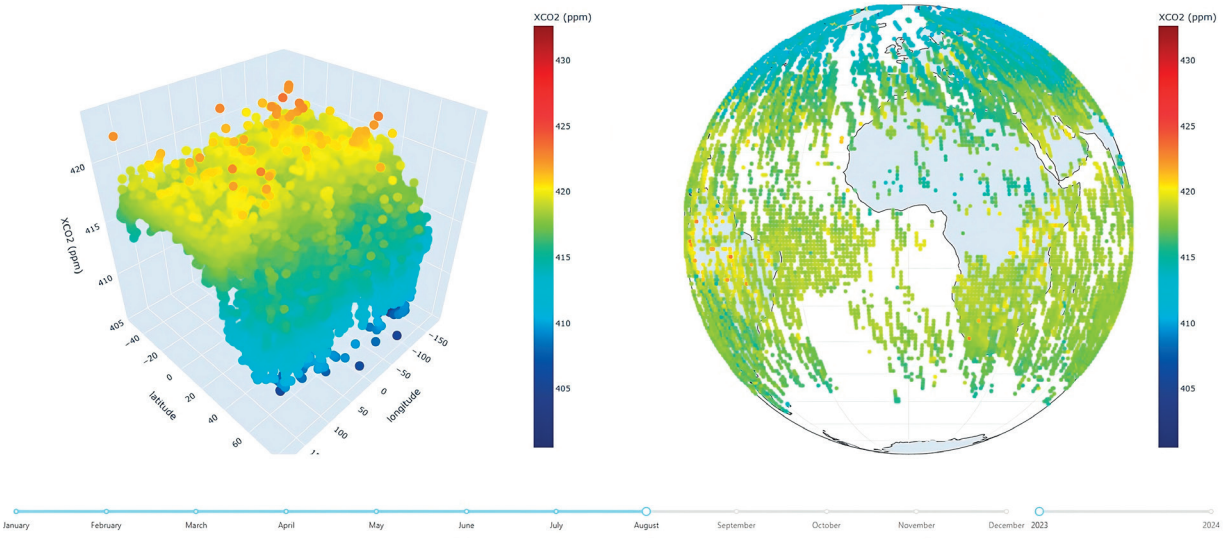

Figure 9 *Graph and visualization for average monthly values - October 2023*

selected datasets. For each selection, the visual outputs are updated dynamically to the chosen time period, enabling direct comparison of spatial and temporal variations in atmospheric CO_2 using both the 3D scatter plots and global map projections.

The left-hand panels of Figures 8 and 9 present the 3D scatter plots of mean XCO_2, the column-averaged dry-air mole fraction of atmospheric CO_2, as a function of latitude (x-axis) and longitude (y-axis) for January 2024 and October 2023 respectively. Each marker corresponds to the average of XCO_2 value per degree of latitude and one degree of longitude. Marker color follows the adjacent color bar (ppm) and encodes the same XCO_2values shown on the vertical axis - warmer colors indicate higher concentrations. Since the plot projects many closely spaced retrievals, regional gradients are best read by following the color and height trends across latitude/longitude ranges rather than individual markers. This representation effectively captures the spatial structure of the XCO_2 field and highlights broad geographic variations.

The right-hand panel shows the corresponding global map projection for the same month and uses the identical color scale and marker values, emphasizing the spatial variability and regional patterns for the same period. Regions without colored symbols correspond to areas where no OCO-2 satellite measurements were available for the selected month. A common color scale (ppm) is applied to both panels to ensure consistent comparison of concentration levels. When considered together, the 3D scatter plots and global map projections provide complementary perspectives on the distribution and magnitude of observed XCO_2. Both panels incorporate interactive functionality that enables detailed exploration of localized spatial patterns. This integrated visualization framework provides a compact, interactive environment for data exploration, anomaly detection, and trend analysis, without the

need for extensive preprocessing or specialized external visualization tools.

5 Discussions

The results indicate that the automated ingestion of OCO-2 Level 2 Standard and Level 2 Lite FP products, facilitated by the OPeNDAP protocol, ensures dependable access to high-resolution atmospheric CO_2data while concurrently minimizing data transfer overhead. This selective data retrieval mechanism, synergistically integrated with the ETL pipeline, decreases the latency between the data availability and its subsequent visualization. In contrast to alternative solutions, such as openEO or SAT-CEP-Monitor, which primarily emphasize interoperability or event processing, the proposed framework distinguishes itself by prioritizing modularity and maintainability within a cohesive cloud environment. Consequently, this design renders the framework suitable for sustained deployment, integration with prospective satellite missions, and the potential for synergistic fusion with ground-based observational data.

The implemented visualization module, built with Dash and Plotly, provides an intuitive interface for the real-time analysis, offering users the ability to examine both global and regional CO_2 patterns. The ability to isolate the CO_2 concentrations over specific territories, demonstrates the adaptability of the system even in regions without open-access in-situ monitoring stations. This highlights the framework's potential to fill the observational gaps and to support environmental research in countries with limited measurement infrastructure.

Despite its advantages, several limitations must be acknowledged. The system's performance depends on the availability and quality of satellite data, which may vary due to orbital coverage, atmospheric conditions, or latency in dataset publication. Furthermore, while the AWS-based deployment ensures scalability, the long-term sustainability could benefit from cost optimization strategies or the adoption of hybrid cloud-edge architectures. The current model also primarily focuses on CO_2 data. Extending it to include other greenhouse gases, such as CH_4 or NO_2, would improve its utility for comprehensive climate monitoring.

6 Conclusion

In this paper are presented the design and implementation of a modular, cloud-based information system for automatic acquisition, processing, and visualization of satellite-derived CO_2 data. The system combines the OCO-2 Level 2 Standard and Level 2 Lite FP products into an ETL pipeline using the OPeNDAP protocol, is deployed on AWS, and offers an interactive web interface for exploring global and regional XCO_2 distributions, including areas without in-situ monitoring.

The main contribution of this study is a unified framework that combines a scalable ETL architecture, modular software design, cloud-native deployment, and advanced visualization. The ETL pipeline extracts only the essential variables from the large NetCDF files, reducing the data transfer and processing time, while maintaining native spatial resolution. A modular, object-oriented design with clearly defined extractor and loader interfaces simplifies testing, maintenance, and adaptation to new data sources or storage back ends. Additionally, a containerized deployment with an interactive Dash/Plotly interface enables the 3D scatter plots and global map projections for detailed analysis of spatial and temporal XCO_2 patterns, even in regions lacking ground observations.

Several limitations must be recognized. The quality and completeness of outputs depend on the availability and latency of OCO-2 products, which are affected by orbital coverage, cloud interference, and publication delays. Relying solely on AWS introduces potential long-term costs and vendor lock-in risks for sustained, high-volume operations. Methodologically, the system currently emphasizes descriptive visualization of CO_2 only, and has not yet integrated the data fusion with other greenhouse gases, meteorological data, or ground-based observations, nor implemented advanced uncertainty quantification.

Future work will expand the framework to include additional satellite missions and trace gases such as CH_4 and NO_2, incorporate external atmospheric and emission models, and systematically validate satellite-based indicators against the in-situ networks. it is also planned to leverage the existing integration points for machine-learning services to enable automated pattern recognition, anomaly detection, and predictive analytics for applications such as transport corridors and industrial regions. Finally, the user experience and computational performance under realistic workloads would be assessed, to further optimize the system for decision support in environmental monitoring, transportation, and industrial informatics.

Acknowledgements

Funded by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under the project No. 09I03-03-V04-00562.

Conflicts of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] SELTENRICH, N. A Satellite view of pollution on the ground: long-term changes in global nitrogen dioxide. *Environmental Health Perspectives* [online]. 2016, **124**(3), A56 [accessed 2025-11-20]. ISSN 0091-6765, eISSN 1552-9924. Available from: https://doi.org/10.1289/ehp.124-A56

[2] HAKKARAINEN, J., IALONGO, I., KOENE, E., SZELĄG, M. E., TAMMINEN, J., KUHLMANN, G., BRUNNER, D. Analyzing local carbon dioxide and nitrogen oxide emissions from space using the divergence method: an application to the synthetic SMARTCARB dataset. *Frontiers in Remote Sensing* [online]. 2022, **3**, 878731 [accessed 2025-11-20]. eISSN 2673-6187. Available from: https://doi.org/10.3389/frsen.2022.878731

[3] KUHLMANN, G., HENNE, S., MEIJER, Y., BRUNNER, D. Quantifying CO_2 emissions of power plants with CO_2 and NO_2 imaging satellites. *Frontiers in Remote Sensing* [online]. 2021, **2**, 689838 [accessed 2025-10-19]. eISSN 2673-6187. Available from: https://doi.org/10.3389/frsen.2021.689838

[4] CAJOVA KANTOVA, N., BELANY, P., HOLUBCIK, M., CAJA, A. Energy consumption depending on the durability of pellets formed from sawdust with an admixture of FFP2 masks. *Energies* [online]. 2022, **15**(13), 4813 [accessed 2025-10-28]. eISSN 1996-1073. Available from: https://doi.org/10.3390/en15134813

[5] LIN, X., VAN DER A, R., DE LAAT, J., ESKES, H., CHEVALLIER, F., CIAIS, P., DENG, Z., GENG, Y., SONG, X., NI, X., HUO, D., DOU, X., LIU, Z. Monitoring and quantifying CO2 emissions of isolated power plants from space. *Atmospheric Chemistry and Physics* [online]. 2023, **23**(11), p. 6599-6611 [accessed 2025-10-15]. ISSN 1680-7375, eISSN 1680-7324. Available from: https://doi.org/10.5194/acp-23-6599-2023

[6] DIMDORE-MILES, O. B., PALMER, P. I., BRUHWILER, L. P. Detecting changes in Arctic methane emissions: limitations of the inter-polar difference of atmospheric mole fractions. *Atmospheric Chemistry and Physics* [online]. 2018, **18**(24), p. 17895-17907 [accessed 2025-10-29]. ISSN 1680-7375, eISSN 1680-7324. Available from: https://doi.org/10.5194/acp-18-17895-2018

[7] BLACKHURST, M., MATTHEWS, H. S. Comparing sources of uncertainty in community greenhouse gas estimation techniques. *Environmental Research Letters* [online]. 2022, **17**(5), 053002 [accessed 2025-10-15]. ISSN 1748-9326. Available from: https://doi.org/10.1088/1748-9326/ac6084

[8] LIU, F., DUNCAN, B. N., KROTKOV, N. A., LAMSAL, L. N., BEIRLE, S., GRIFFIN, D., MCLINDEN, C. A., GOLDBERG, D. L., LU, Z. A methodology to constrain carbon dioxide emissions from coal-fired power plants using satellite observations of co-emitted nitrogen dioxide. *Atmospheric Chemistry and Physics* [online]. 2020, **20**(1), p. 99-116 [accessed 2025-10-15]. ISSN 1680-7375, eISSN 1680-7324. Available from: https://doi.org/10.5194/acp-20-99-2020

[9] WU, S., YAN, Y., DU, Z., ZHANG, F., LIU, R. Spatiotemporal visualization of time-series satellite-derived CO2 flux data using volume rendering and GPU-based interpolation on a cloud-driven digital earth. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* [online]. 2017, **IV-4/W2**, p. 77-85 [accessed 2025-10-11]. ISSN 2194-9042, eISSN 2194-9050. Available from: https://doi.org/10.5194/isprs-annals-IV-4-W2-77-2017

[10] YAVARI, A., MIRZA, I. B., BAGHA, H., KORALA, H., DIA, H., SCIFLEET, P., SARGENT, J., TJUNG, C., SHAFIEI, M. ArtEMon: artificial intelligence and internet of things powered greenhouse gas sensing for real-time emissions monitoring. *Sensors* [online]. 2023, **23**(18), 7971 [accessed 2025-11-05]. eISSN 1424-8220. Available from: https://doi.org/10.3390/s23187971

[11] RUSHTON, C. E., TATE, J. E., SJODIN, A. A modern, flexible cloud-based database and computing service for real-time analysis of vehicle emissions data. *Urban Informatics* [online]. 2025, **4**(1), 1 [accessed 2025-10-17]. eISSN 2731-6963. Available from: https://doi.org/10.1007/s44212-024-00066-4

[12] CAI, K., GUAN, L., LI, S., ZHANG, S., LIU, Y. Full-coverage estimation of CO2 concentrations in China via multisource satellite data and Deep Forest model. *Scientific Data* [online]. 2024, **11**(1), 1231 [accessed 2025-10-29]. eISSN 2052-4463. Available from: https://doi.org/10.1038/s41597-024-04063-9

[13] OTHMAN, A., HRAD, J., HAJEK, J., MAGA, D. Control strategies of hybrid energy harvesting - a survey. *Sustainability* [online]. 2022, **14**(24), 16670 [accessed 2025-10-28]. eISSN 2071-1050. Available from: https://doi.org/10.3390/su142416670

[14] TANIMOTO, H., MATSUNAGA, T., SOMEYA, Y., FUJINAWA, T., OHYAMA, H., MORINO, I., YASHIRO, H., SUGITA, T., INOMATA, S., MÜLLER, A., SAEKI, T., YOSHIDA, Y., NIWA, Y., SAITO, M., NODA, H., YAMASHITA, Y., IKEDA, K., SAIGUSA, N., MACHIDA, T., FREY, M. M., LIM, H., SRIVASTAVA, P., JIN, Y., SHIMIZU, A., NISHIZAWA, T., KANAYA, Y., SEKIYA, T., PATRA, P., TAKIGAWA, M., BISHT, J., KASAI, Y., SATO, T. O. The greenhouse gas observation mission with Global Observing SATellite for Greenhouse gases and Water cycle (GOSAT-GW): objectives, conceptual framework and scientific contributions. *Progress in Earth and Planetary Science* [online]. 2025, **12**(1), 8 [accessed 2025-10-28]. eISSN 2197-4284. Available from: https://doi.org/10.1186/s40645-025-00684-9

[15] JANAIRO, A. G., CONCEPCION, R., GUILLERMO, M., FERNANDO, A. A Cloud computing framework for space farming data analysis. *AgriEngineering* [online]. 2025, **7**(5), 149 [accessed 2025-10-28]. eISSN 2624-7402. Available from: https://doi.org/10.3390/agriengineering7050149

[16] SCHUMACHER, B., GRIFFITHS, P., PEBESMA, E., DRIES, J., JACOB, A., THIEX, D., MOHR, M., BRIESE, C. openEO Platform - showcasing a federated, accessible platform for reproducible large-scale Earth Observation analysis. In: EGU General Assembly 2023: abstracts [online]. 2023. EGU23-8526 [accessed 2025-10-25]. Available from: https://doi.org/10.5194/egusphere-egu23-8526

[17] ROMERO JURE, P. V., CABRAL, J. B., MASUELLI, S. ETL for the integration of remote sensing data. In: Argentine Symposium on Images and Vision / Simposio Argentino de Imagenes y Vision (SAIV 2023) - JAIIO 52: proceedings [online]. 2023 [accessed 2025-10-22]. Available from: http://sedici.unlp.edu.ar/handle/10915/165724

[18] SEMLALI, B.-E. B., AMRANI, C. E., ORTIZ, G., BOUBETA-PUIG, J., GARCIA-DE-PRADO, A. SAT-CEP-monitor: An air quality monitoring software architecture combining complex event processing with satellite remote sensing. *Computers and Electrical Engineering* [online]. 2021, **93**, 107257 [accessed 2025-10-26]. ISSN 0045-7906, eISSN 1879-0755. Available from: https://doi.org/10.1016/j.compeleceng.2021.107257

[19] SEMLALI, B.-E. B., AMRANI, C. E. A Stream processing software for air quality satellite datasets. In: Advanced Intelligent Systems for Sustainable Development (AI2SD'2020). KACPRZYK, J., BALAS, V. E., EZZIYYANI, M. (Eds.). Cham: Springer International Publishing, 2022. ISBN 978-3-030-90633-7, p. 839-853. Available from: https://doi.org/10.1007/978-3-030-90633-7_71

[20] OPeNDAP at American Geophysical Union (AGU) 2024 - OPeNDAP [online] [accessed 2025-11-20]. Available from: https://www.opendap.org/opendap-at-american-geophysical-union-agu-2024/

[21] OPeNDAP at American Geophysical Union (AGU) 2023 - OPeNDAP [online] [accessed 2025-11-20]. Available from: https://www.opendap.org/opendap-at-american-geophysical-union-agu-2023/

[22] University Corporation for Atmospheric Research [online] [accessed 2025-10-27]. Available from: https://www.ucar.edu/