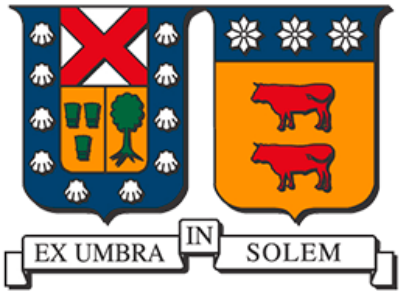
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA



Arquitectura Incremental Open Data Cube Satelital para un Framework N-Dimensional de Geoprocesamiento e Interoperabilidad de Imágenes Multisensor.

Incremental Satellite Open Data Cube Architecture for an N-Dimensional Framework for Multisensor Image Geoprocessing and Interoperability

Enviado en cumplimiento parcial de los requisitos para el grado de

DOCTOR EN INGENIERÍA INFORMÁTICA

por

Roxana Andrea Trujillo Guiñez

Composición de la comisión:

Director de Tesis:	Ph.D. Mauricio Solar R.	UTFSM, Chile.
Evaluador Interno	Ph.D. Julio Sotelo P.	UTFSM, Chile.
Evaluador Externo Nacional	Ph.D. Jaime Ortega P.	Universidad de Chile.
Evaluadora Externa Internacional	Ph.D. Virginia Fernández	U. de la República, Uruguay
Presidente de la comisión	Ph.D. Julio Sotelo P.	UTFSM, Chile.

Marzo 2026

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título **X** Tesis de Postgrado

Título del trabajo: Arquitectura Incremental Open Data Cube Satelital para un Framework N-Dimensional de Geoprocesamiento e Interoperabilidad de Imágenes Multisensor.

Nombre del candidato(a): Roxana Andrea Trujillo Guiñez

Carrera / Grado: Doctorado en Ingeniería Informática

Campus: Casa Central Valparaíso **Departamento:** Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Mauricio Solar, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

X El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 12 de marzo 2026 **Firma:** _____

Estudiante o Candidato(a):

Fecha: 09 de marzo 2026 **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Dedicatoria

Este trabajo está dedicada a mi hijo Leonardo. Su proceso de diálisis y posterior trasplante de riñón transcurrió en paralelo al desarrollo de mis estudios de doctorado, convirtiendo este camino en un desafío profundamente exigente. La fortaleza y resiliencia que demostró en cada etapa fueron una fuente constante de inspiración y me dio la fuerza necesaria para perseverar y concluir esta etapa. Por ello, esta tesis está dedicada a él y a su ejemplo de valentía y esperanza.

ABSTRACT

The rapid growth of Earth Observation (EO) satellite data has highlighted the limitations of traditional execution-centric and file-based processing architectures, particularly for recurrent monitoring scenarios over long satellite time series. While current approaches have improved data access and scalability, support for incremental updates and efficient result reuse in recurrent monitoring scenarios remains limited.

This thesis proposes an **Incremental Multi-Cube architecture** based on an **intelligent orchestration engine** that autonomously manages spatial subdivision, baseline evolution and execution decisions. The orchestrator dynamically partitions areas of interest into stable sub-cubes, optimizes tile selection according to spatial coverage and applies explicit update policies such as incremental appending, temporal gap filling, on-demand sub-cube extraction and controlled recomputation, with adaptive parallelization incorporated as a supporting mechanism. The framework is algorithm and sensor agnostic, enabling the integration of heterogeneous satellite image time series and diverse analytical methods without coupling algorithmic logic to data management or orchestration.

The experimental evaluation, based on multi-year Sentinel-2 time series as a representative EO dataset, demonstrates order of magnitude improvements in execution time and disk I/O compared to classical sequential processing workflows. These results confirm that the architecture efficiently supports recurrent EO monitoring and long-term time-series analysis while avoiding unnecessary recomputation.

RESUMEN

El rápido crecimiento de los datos satelitales de Observación de la Tierra (EO) ha evidenciado las limitaciones de las arquitecturas tradicionales de procesamiento, centradas en la ejecución y en flujos basados en archivos, particularmente en escenarios de monitoreo recurrente sobre largas series temporales de imágenes. Si bien los enfoques actuales han mejorado el acceso a los datos y la escalabilidad, el soporte para actualización incremental y reutilización eficiente de resultados en escenarios de monitoreo recurrente sigue siendo limitado.

Esta tesis propone una **arquitectura Incremental Multi-Cube** basada en un **motor de orquestación inteligente** que gestiona de forma autónoma la subdivisión espacial, la evolución del baseline y las decisiones de ejecución. El orquestador divide dinámicamente las áreas de interés en sub-cubos estables, optimiza la selección de tiles en función de la cobertura espacial y aplica políticas explícitas de actualización, tales como appending incremental, relleno de vacíos temporales, extracción on-demand de sub-cubos y recomputación controlada, incorporando paralelización adaptativa como mecanismo de soporte. El framework es agnóstico a algoritmos y sensores, lo que permite integrar series temporales satelitales heterogéneas y distintos métodos analíticos sin acoplar la lógica algorítmica a la gestión de datos ni a la orquestación.

La evaluación experimental, basada en series temporales multianuales de Sentinel-2 como conjunto EO representativo, evidencia mejoras de orden de magnitud en tiempo de ejecución y en I/O de disco frente a flujos secuenciales clásicos. Estos resultados confirman que la arquitectura soporta de forma eficiente el monitoreo EO recurrente y el análisis de series temporales de largo plazo, evitando recomputaciones innecesarias.

ACKNOWLEDGEMENTS

The proposed framework was evaluated for performance in the context of the project Environmental Hazards Associated with Mining Activities in the Tropics (EDITOR, contract SR/00/413), funded by the Belgian Science Policy (Belspo) STEREO Program. The author acknowledges the Remote Sensing Division of the Centre Spatial de Liège (CSL, Belgium) for hosting her during a research stay, which enabled the performance benchmarking of the framework using dedicated computing resources within this specific validation context. The author also acknowledges Dr. Dominique Derraw for facilitating the research stay at CSL and for the institutional support provided during this period.

The author further acknowledges the methodological contributions presented in prior work, including the event detection algorithm described by Deijns et al., which was employed exclusively during the performance evaluation stage of this research.

TABLE OF CONTENTS

1 Introduction 19

1.1 Context and Research Problem 19

1.2 Hypothesis 21

1.3 Objective 21

1.4 Scope and Limitations of the Research 21

1.5 Document Structure 22

2 Traditional Architectures and Theoretical Foundations 24

2.1 Satellite Images as Multidimensional Earth Observation Data 24

2.1.1 Satellite Image as a Spatial Matrix 29

2.1.2 Multispectral Image as a Third-Order Tensor 31

2.1.3 Temporal Extension: Satellite Image Time Series 32

2.1.4 Multisensor Earth Observation Data 33

2.2 Traditional Processing of Satellite Images 34

2.2.1 Scene-Based and File-Centric Processing Paradigm 35

2.2.2 Sequential Workflows and Reprocessing 38

2.3 Processing Architectures for Large-Scale Earth Observation Data 41

2.3.1 High-Performance and Distributed Processing Pipelines 42

2.3.2 Workflow-Oriented and Cloud-Native Architectures 44

2.3.3 Structural Limitations of Execution-Centric Architectures 46

2.3.4 Transition Toward Persistent and State-Aware Architectures 46

2.4 Data Cubes and Multidimensional Architectures 47

2.4.1 Open Data Cube and Monolithic Approaches 48

2.4.2 Cloud-Native and Chunked Data Approaches 52

2.5 Limitations of Existing Architectures 55

2.6 Need for Incremental and Recurrent Monitoring-Oriented Approaches 56

3 Methodological Proposal 59

3.1 Incremental Multi-Cube Architecture 59

3.2 Construcción de Data Cubes estables 64

3.3 Dynamic Baseline Policies 66

3.4 Orchestrator and Decision Logic 70

3.5 Multi-Sensor Integration Strategy 71

3.5.1 Sensor-Agnostic Architectural Design 71

3.5.2 Master Grid Definition and Spatial Harmonization 72

3.5.3 Resampling Policies and Variable Semantics 74

3.5.4 Temporal Alignment and Incremental Integration 76

3.5.5 Implications for Incremental and Recurrent Monitoring 76

3.6 Adaptive Parallelization and Execution 77

4 Experimental Design and Implementation 80

4.1 Considerations of Implementation 80
4.2 Data Sources Used 81
4.3 Experimental Configuration 83
4.3.1 Study Areas 84
4.3.2 Computational Environment 86
4.4 Use Cases for Architectural Validation 87
4.5 Computational Performance Metrics 88
4.5.1 End-to-End Execution Time 88
4.5.2 CPU Utilization and Concurrency 89
4.5.3 Memory Usage 89
4.5.4 Disk I/O Volume 89
4.5.5 Data Volume and Processing Scale 90
5 Results and Discussion 93
5.1 Scenario 1: Sequential Emulation vs. Adaptive Parallelization 94
5.2 Scenario 2: Sub-Cube Access vs. Full-Cube Recalculation 102
5.3 Architectural Discussion of the Results 107
5.3.1 Execution behavior and adaptive parallelization 107
5.3.2 Rebuild-centric processing versus incremental baseline reuse 108
5.3.3 On-demand sub-cube access as a consequence of persistence 108
5.3.4 Architectural implications and synthesis 109

5.4 Advantages, Limitations, and Trade-offs of the Approach 110

5.4.1 Advantages of the Proposed Approach 110

5.4.2 Architectural Limitations 111

5.4.3 Trade-offs and Design Decisions 112

6 Conclusions and Future Work 115

6.1 General Conclusions 115

6.2 Contributions of the Thesis 116

6.3 Future Work Directions 117

7 BIBLIOGRAPHY 119

LIST OF FIGURES

Figure 2.1 Conceptual illustration of spatial resolution using raster matrices. 25

Figure 2.2 Conceptual illustration of spectral resolution in satellite sensors 26

Figure 2.3 Conceptual illustration of temporal resolution in satellite time series. 27

Figure 2.4 Conceptual illustration of radiometric resolution in satellite sensors. 28

Figure 2.5 Two-dimensional spatial matrix model of satellite imagery 30

Figure 2.6 Multispectral satellite image as a third-order data tensor. 31

Figure 2.7 Tile-based spatial organization of EO data archives. 36

Figure 2.8 Hierarchical structure of scene-based EO archives. 37

Figure 2.9 Sequential file-based EO processing pipeline. 40

Figure 2.10 Distributed batch-processing pipeline architecture. 43

Figure 2.11 Representative execution-centric EO processing architecture. 45

Figure 2.12 Monolithic data cube architecture. 49

Figure 2.13 Principles of cloud-native chunked data models. 54

Figure 3.1 Conceptual overview of the proposed Multi-Cube architecture. 63

Figure 3.2 Regular Zarr single structure. 64

Figure 3.3 Internal workflow of the Multi-Cube incremental engine. 68

Figure 3.4 Geographic coordinate reference system based on latitude and longitude 73

Figure 3.5 Multi-sensor harmonization via reprojection and resampling..... 75

Figure 4.1 Study areas selected to test the framework over a Google Earth basemap. 85

Figure 5.1 Execution concurrency profiles for sequential and parallel modes..... 97

Figure 5.2 Sequential vs. adaptive parallel performance..... 99

Figure 5.3 Framework-driven spatial subdivision of an AOI 101

Figure 5.4 On-demand sub-cube extraction within a study area 102

Figure 5.5 I/O footprints of rebuild-centric and on-demand execution..... 105

Figure 5.6 CPU heatmap for the On-Demand Sub-Cube configuration..... 106

LIST OF TABLES

Table 1 Sentinel-2 spectral bands..... 82

Table 2 Characteristics of the stable cubes generated for the experimental evaluation..... 85

Table 3 Hardware specifications. 86

Table 4 Summary of Performance Metrics 90

Table 5 Main performance summary (sequential vs. parallel)..... 95

Table 6 Comparative summary of performance (Sub-cube)..... 104

LIST OF ACRONYMS AND ABBREVIATIONS

AI — Artificial Intelligence

AOI — Area of Interest

API — Application Programming Interface

AWS — Amazon Web Services

CA — Cellular Automata

CEOS — Committee on Earth Observation Satellites

COG — Cloud Optimized GeoTIFF

CPU — Central Processing Unit

CRC — Cyclic Redundancy Check

CRS — Coordinate Reference System

DAG — Directed Acyclic Graph

EO — Earth Observation

ESA — European Space Agency

FAIR — Findable, Accessible, Interoperable, Reusable

HDF5 — Hierarchical Data Format version 5

HLS — Harmonized Landsat and Sentinel

HPC — High-Performance Computing

HTTP — Hypertext Transfer Protocol

I/O — Input / Output

JPEG2000 — JPEG 2000 image format

LSI-VC — Land Surface Imaging Virtual Constellation

MODIS — Moderate Resolution Imaging Spectroradiometer

MSI — MultiSpectral Instrument

NetCDF — Network Common Data Form

ODC — Open Data Cube

OGC — Open Geospatial Consortium

PID — Process Identifier

RAM — Random Access Memory

SAFE — Standard Archive Format for Europe

SCL — Scene Classification Layer

SHA1 — Secure Hash Algorithm 1

STAC — SpatioTemporal Asset Catalog

UTM — Universal Transverse Mercator

WGS84 — World Geodetic System 1984

WKB — Well-Known Binary

Zarr — Chunked, cloud-native array storage format

SOFTWARE LIBRARIES AND FRAMEWORKS

Dask — Parallel and distributed computing library for Python

xarray — Python library for labeled multidimensional arrays

GeoPySpark — Distributed geospatial processing framework

ArcGIS — Geographic Information System software

CyberGIS — Cyberinfrastructure for Geospatial Information Science

CHAPTER 1

Introduction

1 Introduction

1.1 Context and Research Problem

During the last decade, the availability of Earth Observation (EO) data has grown exponentially, driven by the deployment of satellite constellations offering medium-to-high spatial resolution, high temporal revisit frequency, and near-global coverage. This trend has been strongly shaped by long-term operational missions such as Landsat, which has provided consistent observations for more than four decades (Roy et al., 2014), and Sentinel-2, whose high revisit rate and multispectral capabilities have enabled the construction of dense, continuous time series suitable for environmental monitoring and change detection (Drusch et al., 2012; Wulder et al., 2019). These multi-year EO archives now constitute a fundamental basis for analyzing environmental and territorial processes and for systematically monitoring dynamic phenomena across multiple spatial scales.

In parallel with this data growth, a substantial portion of recent developments in EO infrastructures has focused on improving efficiency in data management, access, and storage. For instance, cloud-native formats based on chunked multidimensional arrays, such as Zarr, have been proposed to enable scalable access to large EO datasets through object storage while supporting parallel I/O and lazy evaluation (Abernathey et al., 2021). Similarly, cloud-optimized raster formats such as COG (OGC Cloud Optimized GeoTIFF Standard, 2023), together with standardized discovery mechanisms like STAC catalogs (STAC Specification, 2021), have facilitated direct, on-demand access to satellite data without requiring full local downloads (Baumann et al., 2018). These technological advances underpin the emergence of data-cube-based infrastructures, including national and continental-scale EO data cubes, which aim to deliver FAIR and analysis-ready datasets to a broad user community (Giuliani et al., 2017; Killough, 2018).

Despite these advances, the architectural emphasis of most existing EO platforms remains strongly oriented toward repository-level efficiency, prioritizing how data are stored, indexed, and retrieved. As noted by Giuliani et al. (2019), improvements in access performance and data standardization do not necessarily translate into equivalent progress in the coordination of analytical workflows that operate on evolving EO time series. Recent

reviews of cloud-native geospatial analytics further highlight that, while scalable storage and distributed processing are increasingly well supported, limited attention has been paid to how analytical processes should adapt as new observations are continuously incorporated (Sun et al., 2023).

This limitation is particularly relevant because satellite time-series processing inherently involves non-trivial decisions related to spatial, spectral and temporal data selection, the reuse of previously generated products, and the consistent integration of new observations. In many current approaches, such decisions are not explicitly formalized at the architectural level; instead, they are embedded in specific implementations, ingestion pipelines or ad hoc scripts, as observed in both file centric workflows and cube-based processing systems (Appel and Pebesma, 2019; Giuliani et al., 2019). As a consequence, the correct operation of EO analysis workflows often relies on specialized technical expertise, effectively transferring computational and architectural complexity to end users whose primary objectives are typically focused on data analysis and interpretation.

Moreover, although several frameworks have demonstrated notable gains in storage efficiency and scalable access, they continue to exhibit limitations in the intelligent orchestration of processes operating over dynamic and multisensor EO time series. Dritsas and Trigka (2025), for example, emphasize that current systems often lack mechanisms for automatically deciding when to reuse existing results, when to incrementally update time series, and when full reprocessing is unavoidable. The absence of such decision-making capabilities complicates the management of recurrent monitoring scenarios and frequently leads to unnecessary recomputation and suboptimal utilization of computational resources, particularly in long-term operational settings.

In this context, the research problem addressed by this thesis arises from the lack of computational architectures that natively integrate efficient large-scale satellite data management with decision-driven orchestration mechanisms capable of operating coherently under spatial, temporal, and multisensor complexity. There is a clear need for frameworks that can sustain recurrent analysis and monitoring workflows while delivering analysis-ready EO data to users, without requiring detailed manual control of underlying processes or the continuous intervention of domain specialists. Addressing this gap requires moving beyond

repository centric designs toward architectures in which operational decisions, such as reuse, incremental updating and execution strategies, are intrinsic components of the system rather than external responsibilities delegated to users or deployment specific configurations.

1.2 Hypothesis

In alignment with the identified research problem and the gaps in traditional satellite time series architectures, the following general hypothesis is formulated:

The implementation of an adaptive framework based on multidimensional data structures, integrating a decision and orchestration engine that autonomously manages data flow and updates, increases efficiency in satellite time series analysis by reducing computational time and storage costs compared to global recalculation schemes.

It is argued that the integration of embedded operational intelligence and spatially disaggregated data cubes is the key architectural mechanism for overcoming the structural limitations of traditional architectures.

1.3 Objective

The objective of this thesis is to design and validate a computational architecture for processing satellite time series that integrates efficient management of large volumes of Earth Observation data, based on data-cube structures, with process orchestration mechanisms capable of automatically making decisions in response to spatial, spectral, temporal, and multisensor complexity. The proposed architecture is oriented toward supporting recurrent analysis and monitoring workflows, providing users with analysis-ready data for interpretation without requiring detailed manual control of the underlying processes.

In addressing these objectives, this thesis contributes the design, implementation and empirical validation of an incremental, policy-driven EO data cube architecture that enables efficient reuse of large spatio-temporal archives under recurrent monitoring scenarios.

1.4 Scope and Limitations of the Research

This thesis focuses on the design, implementation and validation of a computational

architecture for large-scale satellite time-series processing, with emphasis on efficient data management and intelligent orchestration of processes operating on EO imagery. The scope of the study includes the definition of the framework structure, the mechanisms for incremental construction and updating of data cubes and the decision policies required to sustain recurrent analysis and monitoring workflows in an efficient and reproducible manner.

The validation of the proposed approach is conducted from an architectural and computational perspective, considering metrics related to the framework's behavior, such as processing efficiency, reuse of previously generated data, and reduction of unnecessary reprocessing.

This thesis does not aim to propose or evaluate specific algorithms for the analysis or interpretation of satellite imagery, nor to compare the performance of particular analytical methods. Likewise, the optimization of thematic models and the evaluation of final products from an applied or domain specific perspective are outside the scope of this research. Decisions related to the design of analytical algorithms, variable selection, and thematic interpretation of results are therefore considered beyond the objectives of this study and are viewed as subsequent stages that may benefit from the proposed framework.

1.5 Document Structure

This document is organized into six chapters. **Chapter 1** introduces the general context of the research, formulates the problem addressed, defines the objective of the thesis, and establishes its scope and limitations. **Chapter 2** presents the state of the art and the theoretical foundations related to Earth Observation time-series processing and the computational architectures used for their management. **Chapter 3** describes the methodological proposal of the thesis, detailing the framework architecture and the mechanisms for process orchestration and incremental data management. **Chapter 4** addresses the implementation aspects and the experimental design used to validate the proposed architecture. **Chapter 5** presents and discusses the results obtained, analyzing the framework's behavior under different scenarios. Finally, **Chapter 6** outlines the main conclusions of the work and identifies directions for future research.

CHAPTER 2

Traditional architectures and theoretical foundations

2 Traditional Architectures and Theoretical Foundations

The analysis and processing of EO time series have been addressed through a wide range of computational approaches, which have progressively evolved in response to the sustained growth in data volume, sensor diversity and acquisition frequency of satellite imagery (Fu et al., 2024; Durbha et al., 2023; Chen et al., 2024; Selea, 2023). This chapter presents a structured review of the main theoretical foundations and traditional architectures employed in EO data processing, with particular emphasis on sequential workflows, file-centric processing paradigms and multidimensional approaches proposed for managing large-scale satellite data archives.

Building upon this review, the chapter examines the key limitations of existing approaches when confronted with recurrent monitoring scenarios and the continuous extension of satellite time series. This analysis establishes the conceptual framework that motivates the need for incremental, architecture-aware processing strategies and explicit process orchestration mechanisms. In doing so, it provides the theoretical context required to understand and justify the methodological proposal introduced in the subsequent chapters of this thesis.

2.1 Satellite Images as Multidimensional Earth Observation Data

A satellite image can be understood as a discrete representation of electromagnetic energy reflected or emitted by the Earth's surface and atmosphere, as measured by a spaceborne sensor. This measurement process involves sampling a fundamentally continuous physical signal across multiple dimensions, including space, spectrum and time. Each pixel in a satellite image corresponds to an integrated observation over a finite ground area, a specific spectral interval and a given acquisition time, as determined by the sensor's optical design, orbit, and acquisition geometry (Richards, 2013; Jensen, 2016).

From a computational perspective, satellite imagery constitutes a structured and multidimensional sampling of a continuous radiometric field, constrained by the sensor's spatial resolution, spectral response functions, temporal revisit cycle and radiometric sensitivity. The resulting data are stored as discrete digital values that approximate the

underlying physical signal through quantization and interpolation processes. These constraints directly influence the level of detail, accuracy and comparability of satellite observations, particularly when integrating multi-temporal or multisensor datasets. Understanding satellite images as discretized samples of a continuous spatio-spectral-temporal signal provides the conceptual foundation for treating EO data as multidimensional arrays or tensors, which is a prerequisite for the construction and analysis of data cube architectures (Schowengerdt, 2007; Mahecha et al., 2020).

Each satellite observation is characterized by four fundamental resolution dimensions:

- **Spatial resolution**, which defines the ground area represented by each pixel and determines the spatial sampling density of the image

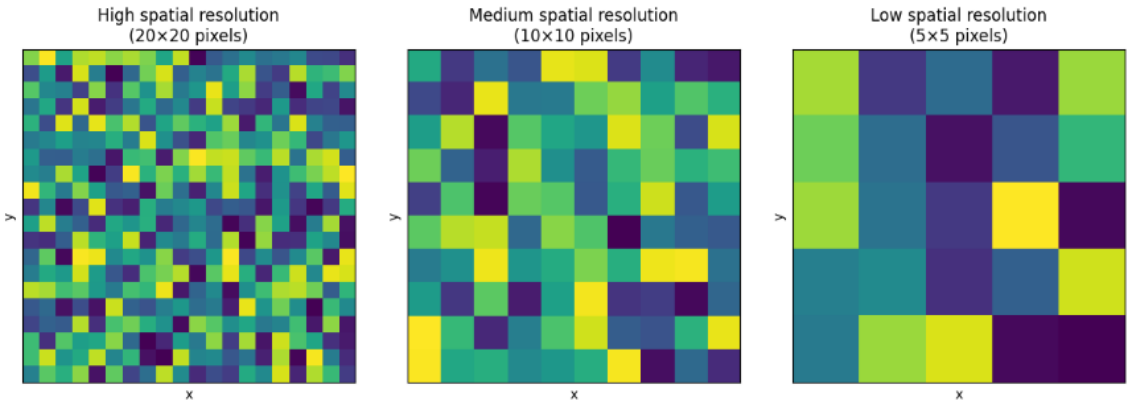


Figure 2.1 Conceptual illustration of spatial resolution using raster matrices.
Conceptual illustration of spatial resolution in raster-based Earth observation data. The same spatial domain is represented as a two-dimensional matrix at three different spatial resolutions: high spatial resolution, where small pixels capture fine-scale spatial variability; medium spatial resolution, where surface heterogeneity is partially aggregated and low spatial resolution, where large pixels represent averaged values over broader ground areas.

- **Spectral resolution**, which describes the number, position and width of spectral bands used to sample the electromagnetic spectrum.

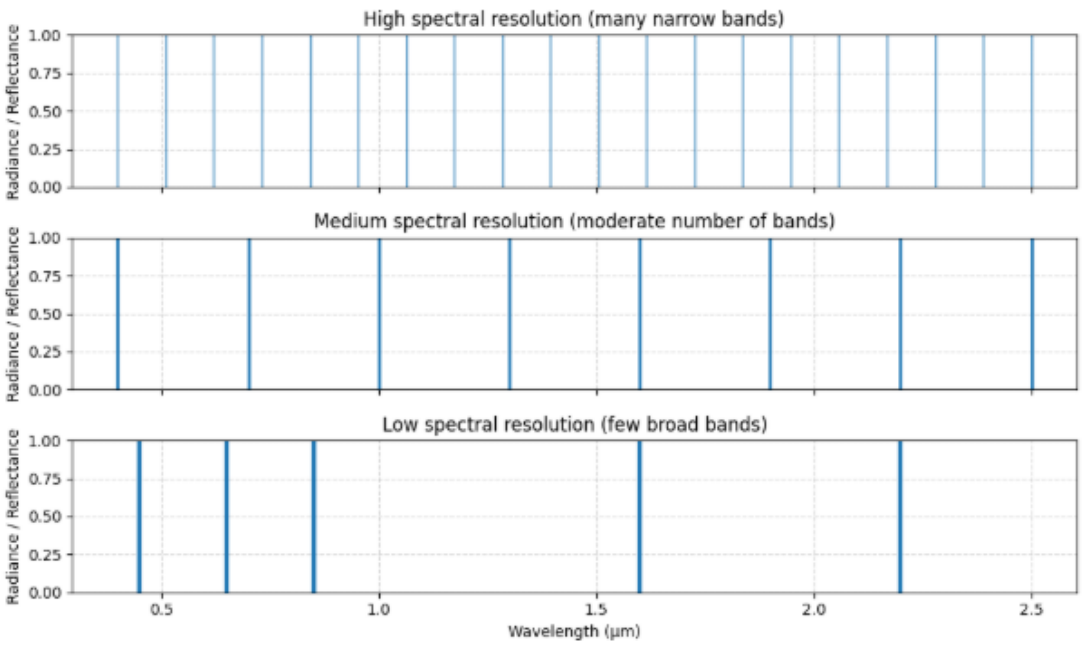


Figure 2.2 Conceptual illustration of spectral resolution in satellite sensors

The same portion of the electromagnetic spectrum is sampled using different spectral configurations: high spectral resolution, characterized by many narrow spectral bands; medium spectral resolution, with a moderate number of bands; and low spectral resolution, where few broad bands integrate radiance over wider wavelength intervals. Spectral resolution determines the ability of a sensor to discriminate between surface materials with similar spatial characteristics but distinct spectral signatures.

- **Temporal resolution**, which refers to the frequency at which the same geographic location is revisited and observed over time.

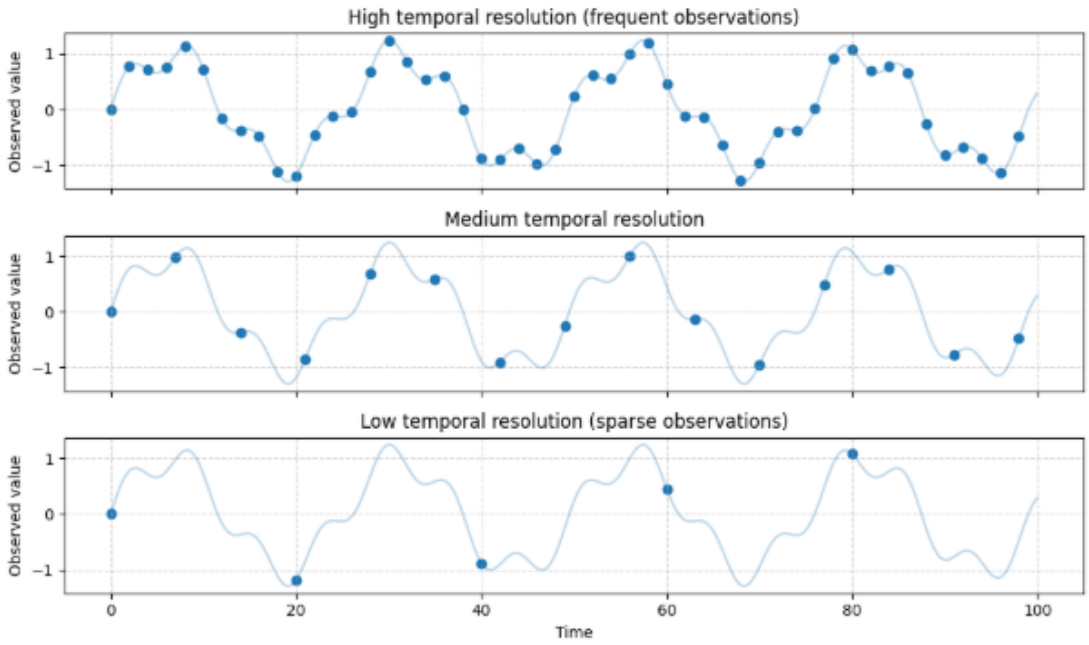


Figure 2.3 Conceptual illustration of temporal resolution in satellite time series. The same underlying temporal signal is sampled at different observation frequencies: high temporal resolution, with frequent acquisitions capturing short-term variability; medium temporal resolution, where the main temporal trends are preserved and low temporal resolution, characterized by sparse observations that may miss rapid changes or transient events. Temporal resolution determines the ability of a satellite time series to represent the dynamics of surface processes over time.

- **Radiometric resolution**, which specifies the number of discrete intensity levels used to encode the measured signal.

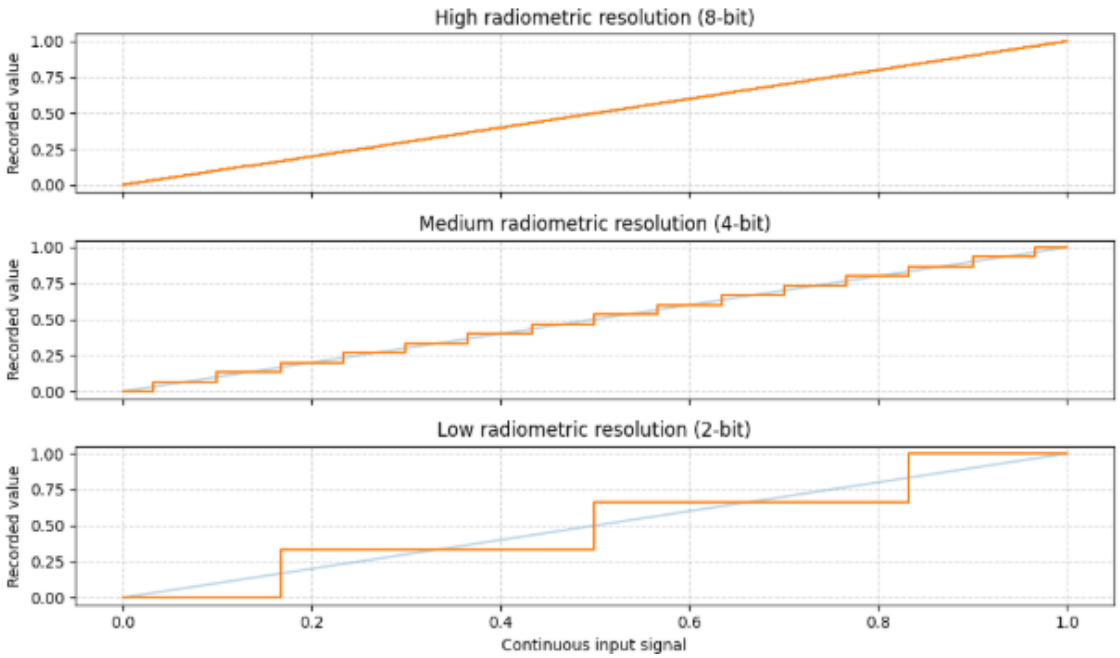


Figure 2.4 Conceptual illustration of radiometric resolution in satellite sensors. The same continuous physical signal is recorded using different radiometric resolutions: high radiometric resolution, where a large number of digital levels allows fine discrimination of radiance or reflectance values; medium radiometric resolution, with coarser quantization and low radiometric resolution, where few digital levels result in strong discretization effects. Radiometric resolution determines the sensitivity of a sensor to subtle differences in measured energy.

These resolution dimensions jointly define the structure and information content of EO data and directly influence how satellite imagery can be stored, processed and analyzed. As EO archives grow in spatial extent, temporal depth and sensor diversity, these dimensions naturally lead to multidimensional data representations rather than independent two-dimensional images.

In the following subsections, satellite imagery is progressively formalized from a two-dimensional spatial matrix to higher-order tensor representations that explicitly capture spectral, temporal and multisensor complexity. This formalization provides the mathematical foundation required to reason about processing strategies and architectural design choices addressed throughout the remainder of this thesis.

While satellite imagery is often treated operationally as a collection of independent raster files, its intrinsic structure is significantly richer. At increasing levels of abstraction, satellite data evolve from two dimensional spatial matrices to multispectral, spatio-temporal and multisensor data constructs. This section formalizes these representations, establishing the theoretical foundation required to reason about processing strategies and architectural design choices addressed in subsequent sections.

2.1.1 Satellite Image as a Spatial Matrix

At its most basic level, a satellite image can be represented as a two-dimensional spatial matrix, where each element corresponds to a discrete measurement recorded over a specific ground location. Formally, a single-band satellite image is defined as:

$$I \in \mathbb{R}^{\{H \times W\}} \qquad (2.1)$$

where H and W denote the number of rows and columns of the spatial grid, respectively. Each matrix element $I[i, j]$ represents the radiometrically calibrated value associated with the pixel located at row i and column j. This value corresponds to a physical quantity measured by the sensor, such as radiance, surface reflectance, brightness temperature or radar backscatter, depending on the sensor type and processing level (Lillesand et al., 2015). Each matrix element represents the aggregated sensor response over a finite ground area, defined by the spatial resolution of the acquisition and constitutes the fundamental spatial sampling unit of satellite imagery.

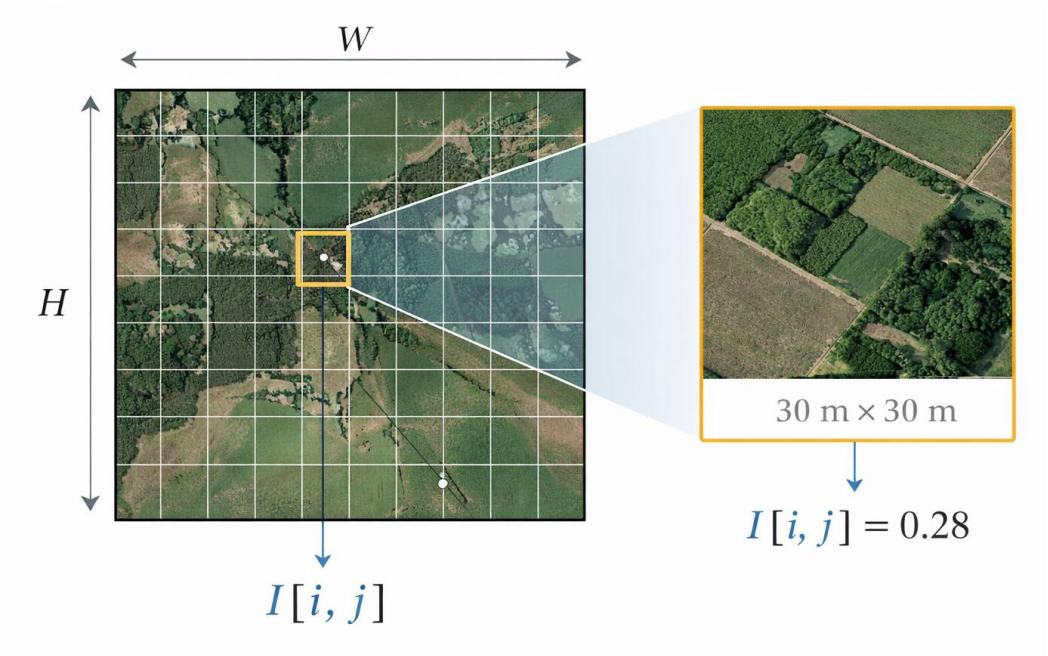


Figure 2.5 Two-dimensional spatial matrix model of satellite imagery

Conceptual representation of a satellite image as a two-dimensional spatial matrix with H rows and W columns. Each matrix element I[i, j] corresponds to a pixel representing the aggregated sensor response over a finite ground area defined by the spatial resolution of the acquisition. In this example, the highlighted pixel has a spatial resolution of 30 m × 30 m, and its associated matrix value I[i, j] = 0.28 represents a radiometrically calibrated measurement, such as surface reflectance, stored in the image matrix.

This matrix-based representation underpins most EO products distributed in standard georeferenced raster and array-based formats, such as GeoTIFF and NetCDF, which store spatial data together with essential metadata for geolocation and interpretation (Lillesand et al., 2015). The spatial resolution of the image determines the physical area represented by each matrix element, directly linking the computational representation to real-world spatial scales. Consequently, spatial operations such as filtering, resampling, neighborhood analysis, and spatial aggregation can be interpreted as matrix operations applied over I.

From a computational standpoint, representing satellite imagery as a spatial matrix enables the application of well-established image processing and numerical analysis techniques. However, it also introduces constraints related to memory usage and data locality, particularly

when large spatial extents or high-resolution imagery are involved. These constraints become increasingly relevant as spatial matrices are combined into larger analytical constructs, such as mosaics or long time series.

2.1.2 Multispectral Image as a Third-Order Tensor

Most contemporary EO missions acquire measurements across multiple spectral bands, capturing information at different wavelengths that correspond to distinct physical properties of the Earth's surface. As a result, satellite images are inherently multispectral and are more accurately modeled as third-order tensors rather than simple matrices. This representation is expressed as:

$$I \in \mathbb{R}^{\{H \times W \times B\}} \qquad (2.2)$$

where B denotes the number of spectral bands measured by the sensor. In this formulation, each element $I[i, j, b]$ represents the measurement recorded at spatial location (i, j) in spectral band b.

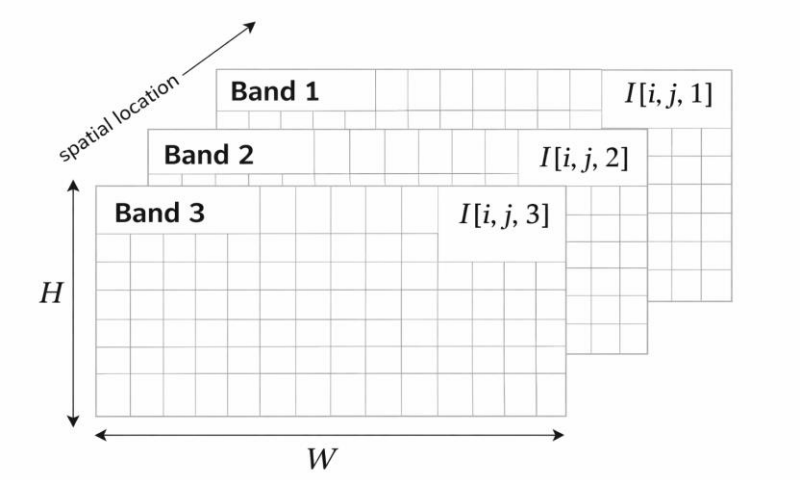


Figure 2.6 Multispectral satellite image as a third-order data tensor.
Conceptual representation of a multispectral satellite image modeled as a third-order tensor of size H × W × B. Each spectral band is represented as a two-dimensional spatial matrix with identical spatial dimensions (H, W), and the stacked matrices together form the spectral dimension B. In this formulation, each element I[i, j, b] corresponds to the radiometrically calibrated measurement recorded at spatial location (i, j) for three spectral bands.

This tensorial representation explicitly captures the spectral dimension of EO data and provides the mathematical basis for a wide range of analytical operations. Spectral transformations, band combinations and the computation of derived variables, such as vegetation, water or built-up indices, are defined as functions operating along the spectral axis of this tensor (Gómez-Chova et al., 2015).

From an architectural perspective, the introduction of the spectral dimension significantly increases data volume and computational complexity. Even moderate spatial resolutions combined with multiple spectral bands result in large multidimensional arrays. Traditional processing workflows often handle this complexity implicitly, processing each band separately or storing bands as independent files, without explicitly acknowledging the underlying tensor structure. This disconnects between data representation and processing logic becomes increasingly problematic as datasets grow in size and heterogeneity.

2.1.3 Temporal Extension: Satellite Image Time Series

A defining characteristic of EO systems is their ability to repeatedly observe the same geographic areas over time, enabling the analysis of temporal dynamics, seasonal patterns, and long-term trends. A satellite image time series can be formally defined as an ordered collection of images acquired at different times:

$$\mathcal{I} = \{I_t \mid t \in T\} \qquad (2.3)$$

where $T = \{t_1, t_2, \ldots, t_m\}$ denotes the set of acquisition times. Each image I_t preserves the spatial and spectral structure defined in Equations (2.1) and (2.2), while capturing temporal variability arising from environmental processes or abrupt disturbances (Verbesselt et al., 2010; Zhu & Woodcock, 2014).

For computational and architectural analysis, this collection can be equivalently represented as a fourth-order tensor:

$$\mathcal{I} \in \mathbb{R}^{\{T \times H \times W \times B\}} \qquad (2.4)$$

In this representation, each element $\mathcal{I}[t, i, j, b]$ corresponds to the observation at time t, spatial location (i, j), and spectral band b. This formulation makes explicit the full spatio-temporal and spectral dimensionality of satellite image time series and forms the conceptual foundation for multidimensional data cube abstractions.

While mathematically compact, this representation poses significant challenges for traditional processing workflows. As the temporal dimension grows, reprocessing the entire tensor upon the arrival of new observations results in substantial redundant computation, excessive disk input/output (I/O) operations and limited scalability. These issues are particularly acute in recurrent monitoring scenarios, where analyses are repeatedly performed over fixed spatial regions.

2.1.4 Multisensor Earth Observation Data

Modern EO archives are rarely limited to a single satellite mission. Instead, they integrate observations from multiple sensors, each characterized by distinct spatial resolutions, spectral configurations, temporal sampling rates and measurement principles (Fang et al., 2023). Recent efforts have addressed these challenges through the generation of harmonized multisensor surface reflectance products, such as the Landsat–Sentinel Harmonized Dataset (HLS), which explicitly align radiometric, geometric and temporal characteristics to support consistent multisensor time-series analysis at scale (Ju et al., 2025).

A multisensor EO dataset ($\mathcal{D}$) can therefore be formalized as a collection of sensor-specific spatio-temporal tensors:

$$\mathcal{D} = \mathcal{I}^{\{(s)\}} \qquad (2.5)$$

where S denotes the set of sensors and each tensor $\mathcal{I}\{(s)\}$, for sensor s, is defined as:

$$\mathcal{I}^{\{(s)\}} \in \mathbb{R}^{\{T_s \times H_s \times W_s \times B_s\}} \qquad (2.6)$$

This formulation explicitly captures heterogeneity across sensors and highlights a fundamental limitation of traditional EO processing paradigms: the absence of a unified abstraction capable of coherently managing heterogeneous data structures. Differences in spatial resolution, temporal coverage, spectral content and preprocessing requirements must be reconciled manually or through ad hoc workflows, increasing system complexity and reducing reproducibility (Mahecha et al., 2020), often require harmonization strategies to enable consistent multisensor time-series analysis (Claverie et al., 2018).

The multisensor formulation introduced in Equation (2.5) emphasizes that EO data archives are not merely large, but structurally complex. Traditional EO processing paradigms emerged in the context of long-running optical satellite missions designed to provide consistent, global observations over extended periods, laying the foundation for large-scale image archives and scene-based analytical workflows (Roy et al., 2014). This complexity directly motivates the need for processing architectures that explicitly acknowledge data dimensionality, heterogeneity and temporal evolution, issues that are examined in the following sections of this chapter.

2.2 Traditional Processing of Satellite Images

Traditional satellite image processing has historically been organized around sequential and sensor-centric workflows, in which each image acquisition is treated as an independent processing unit. Under this paradigm, raw or minimally processed satellite scenes are ingested individually, corrected, transformed and analyzed through a predefined chain of operations, typically executed in a fixed order. These workflows were originally designed in a context where data volumes were relatively moderate, revisit times were longer and analyses were often performed on a per-scene or per-campaign basis rather than through continuous temporal monitoring.

Conventional processing pipelines usually follow a linear sequence of steps that include radiometric and geometric corrections, atmospheric compensation, reprojection, resampling, and the derivation of thematic products or spectral indices. Each stage produces intermediate outputs that are commonly stored as separate files and subsequent processing stages operate by reading and writing these files explicitly. As a result, the overall workflow is strongly

coupled to scene-based data representations and local storage structures, reflecting the dominant computational and data-management practices at the time these approaches were established.

This shift from scene-based analysis toward continuous land surface monitoring has been widely discussed in the EO literature, emphasizing the need for time-series–oriented processing paradigms (Wulder et al., 2019).

While traditional processing strategies have proven effective for many applications, particularly in exploratory analysis and small to medium scale studies, their design assumptions impose important constraints when dealing with long satellite image time series, multisensor datasets and recurrent monitoring scenarios. The following sections examine the characteristics of these sequential and file-centric workflows in more detail, highlighting the structural limitations that motivate the transition toward more scalable and incremental processing architectures.

2.2.1 Scene-Based and File-Centric Processing Paradigm

Most current EO data infrastructures are organized around a scene-based and file-centric processing paradigm, in which each satellite acquisition is treated as an independent data product. Under this model, images are distributed as self-contained files or archives, commonly referred to as scenes, tiles or granules, that encapsulate all spectral bands and metadata associated with a single acquisition time and spatial footprint. This organizational logic emerged alongside early EO missions, reflecting historical constraints in storage capacity, data transfer and computational resources, as documented in classical remote sensing literature (Lillesand et al., 2015). In practice, this paradigm is implemented through fixed spatial partitioning schemes, in which the Earth's surface is subdivided into discrete spatial units that define the fundamental processing entities of traditional EO data archives, where each tile is treated as an independent processing entity as illustrated in **Figure 2.7**

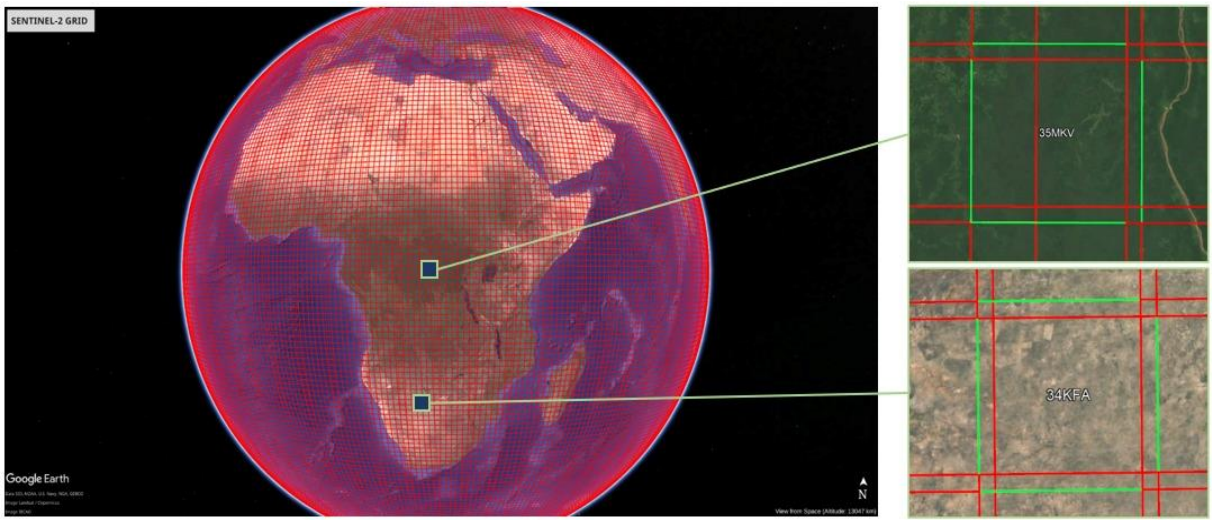


Figure 2.7 Tile-based spatial organization of EO data archives.
Example of a scene- and tile-based spatial partitioning scheme used in traditional EO data archives. The Earth's surface is subdivided into fixed spatial units (tiles or granules), each representing an independent image product associated with a specific spatial footprint and acquisition geometry. These tiles constitute the fundamental processing entities in scene-based and file-centric workflows. Source: Google Earth Basemap, Sentinel-2 grid.

The scene-based approach remains widely adopted across optical EO missions, ranging from long-running public programs such as Landsat to more recent constellations including Sentinel-2 and MODIS. As highlighted by Woodcock et al. (2008), the availability of standardized scene products enabled broad access to satellite imagery but also reinforced a processing model centered on discrete acquisitions. Within this context, user workflows typically follow a request-driven pattern in which an area of interest and a temporal range are specified, relevant image products are identified through catalogs or application programming interfaces, and individual scenes are subsequently processed in isolation. Although sensor characteristics and file formats vary between missions (Reuter et al., 2015), the underlying logic of treating each acquisition as a standalone processing unit remains largely consistent.

A defining characteristic of the file-centric paradigm is that processing is tightly coupled to discrete image files rather than to persistent spatiotemporal data structures. Intermediate products are explicitly generated and stored at multiple stages of the workflow, resulting in repeated disk input/output operations. Gómez-Chova et al. (2015) note that this design leads

to redundant execution of identical preprocessing steps, such as reprojection, resampling or masking, when overlapping spatial or temporal requests are issued, even if the same data have already been processed in previous analyses (CEOS LSI-VC, 2024)

In traditional EO processing systems, the execution of reference workflows is tightly coupled to the physical organization of data on disk. Scene or tile-based products are stored using rigid hierarchical directory structures that reflect spatial partitioning schemes, acquisition geometry and processing levels. This organization directly conditions how processing modules access input data and materializes intermediate results, reinforcing the file-centric nature of the workflow. An example of this rigid on-disk organization is shown in **Figure 2.8**, where a catalog-driven hierarchy progressively narrows from a mission-level collection to UTM zone, sub-zone, and finally individual tile packages (compressed archives) that act as the atomic units for retrieval and processing.

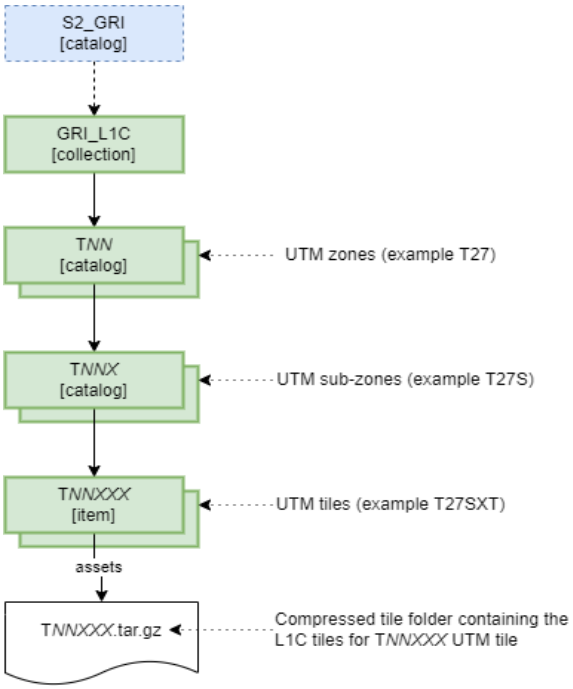


Figure 2.8 Hierarchical structure of scene-based EO archives.
Example of a hierarchical organization used in scene/tile-based EO archives. Products are indexed from a collection level down to UTM zones, sub-zones, and individual tile identifiers, which are commonly distributed as compressed packages containing the tile assets. This structure reflects how many file-centric pipelines locate inputs and materialize intermediate outputs. Source: Sentinel-2 Level-1C product specification (Copernicus Programme).

This kind of hierarchy makes data access highly deterministic, but it also hard-codes file-level assumptions into processing logic, which reinforces repeated reads/writes and scene-by-scene execution.

Such organizational choices also promote sequential and monolithic workflows, in which scenes are processed independently and later combined through mosaicking, stacking, or temporal aggregation operations. While such strategies are effective for event-based analyses or short temporal studies, their limitations become apparent when applied to long-term monitoring or multisensor integration. For example, ensuring temporal consistency across independently processed acquisitions often requires additional harmonization efforts, particularly when processing baselines evolve over time, as observed in large-scale initiatives integrating Landsat and Sentinel-2 data (Claverie et al., 2018).

Despite recent advances in data access mechanisms, such as cloud-hosted archives, catalog standards, and optimized file formats, the fundamental logic of scene-based and file-centric processing has largely persisted. Consequently, most operational EO pipelines remain ill-suited to support incremental data growth, baseline reuse or continuous re-analysis over stable spatial domains. These structural limitations motivate the development of alternative architectures that decouple analytical workflows from individual files and instead operate over persistent, incrementally maintained spatiotemporal representations.

2.2.2 Sequential Workflows and Reprocessing

Traditional EO processing systems are commonly implemented through reference-based workflows, in which each satellite image product is processed independently following a predefined and sequential chain of operations. This execution model defines a canonical processing path that transforms scene, tile or granule-based inputs into analysis-ready products, ensuring reproducibility and consistency across processing runs.

At a high level, a traditional reference workflow comprises the following core processing stages:

- **Ingestion of scene-based products**, where each acquisition is treated as an independent processing unit associated with a fixed spatial footprint and acquisition time.

- **Radiometric and geometric preprocessing**, including calibration, correction, reprojection and resampling operations applied separately to each scene or tile

- **Quality masking and spatial subsetting**, used to remove invalid observations and restrict processing to user-defined areas of interest.

- **Intermediate storage of processed outputs**, where results from each stage are written as standalone files and reused only implicitly.

- **Temporal aggregation and analysis**, performed a posteriori by stacking or mosaicking independently processed scenes to construct time series or composite products.

While this sequential structure is conceptually simple and operationally robust, it does not account for spatial or temporal overlap between processing requests, a limitation that becomes particularly evident in long EO time series analysis (Simões et al., 2021; Tang et al., 2024). Each execution is treated as an isolated run, and identical preprocessing steps are repeatedly applied even when the underlying data have already been processed in previous analyses.

Figure 2.9 illustrates a representative traditional reference processing workflow, highlighting both its sequential execution logic and its modular organization. Within this paradigm, EO data are handled as discrete files that traverse successive processing stages, with intermediate products explicitly materialized at each step, resulting in a strong coupling between analytical workflows and file-based data representations supported by local or distributed storage systems.

In operational implementations, this reference workflow is commonly realized through a modular architecture in which responsibilities are intentionally separated across functional components: interactive discovery and query formulation are managed by interface or API

layers, while CPU-intensive preprocessing tasks such as decoding, masking, resampling, and clipping, are delegated to dedicated processing modules. Time series assembly and aggregation are performed by analysis components, and final product generation is confined to downstream output modules.

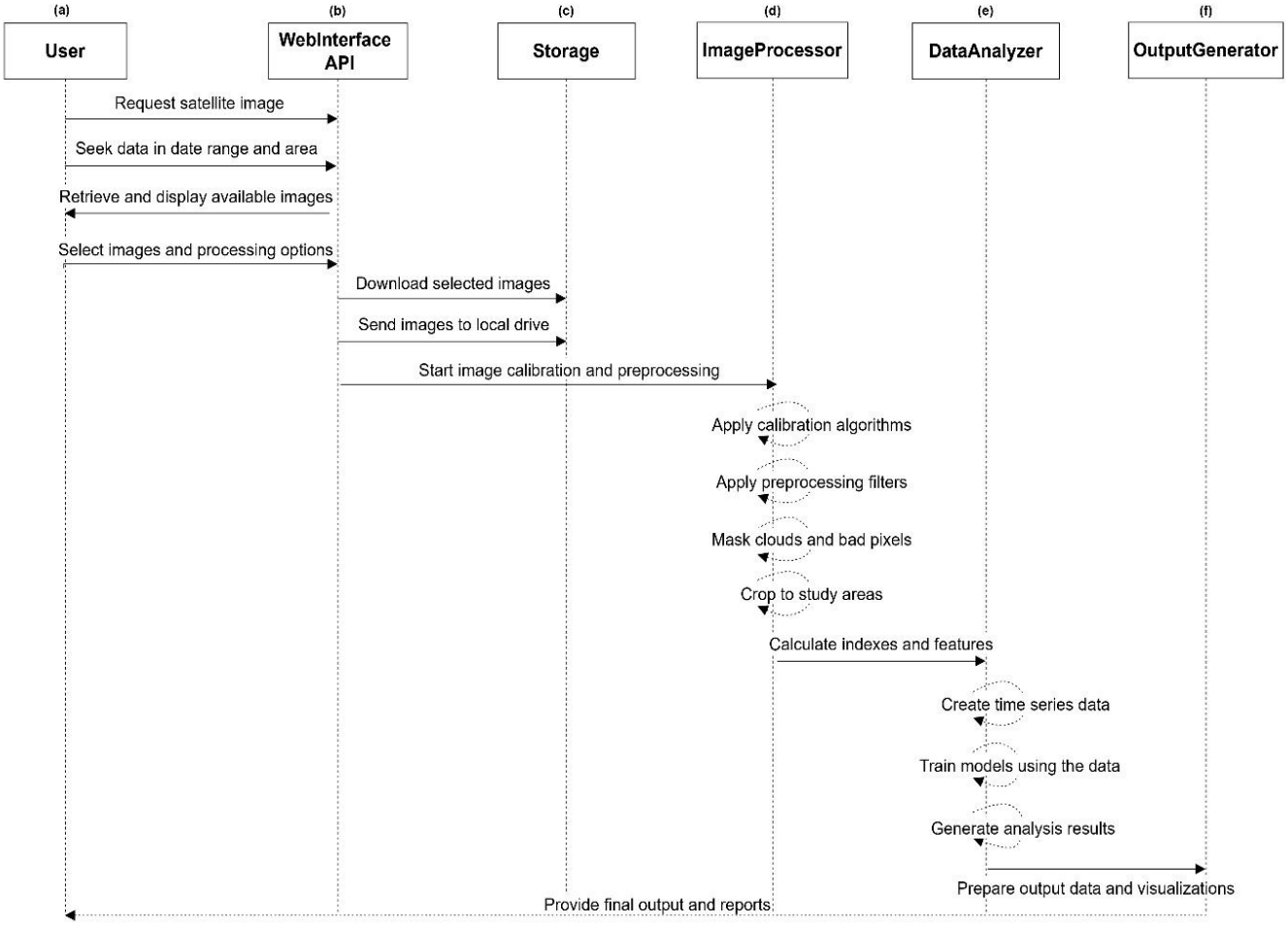


Figure 2.9 Sequential file-based EO processing pipeline.

The figure summarizes a conventional pipeline in which each user request independently triggers the same end-to-end stages: (a) user request definition, including the area of interest, temporal range, and processing parameters; (b) data discovery through a web interface or API via catalog queries and scene selection; (c) downloading and local storage of selected products; (d) calibration and preprocessing steps, such as atmospheric correction, masking, and spatial cropping; (e) an analysis stage, including time series construction and feature or index computation; and (f) generation of final outputs, such as analytical results, visualizations, and reports.

Despite this modularization, the underlying execution model remains reference-based and file-centric, and significant performance bottlenecks tend to arise at module boundaries (Enache et al., 2023). These bottlenecks are commonly associated with bandwidth-limited data transfers between interface layers and storage, repeated decoding and disk I/O during preprocessing, and the consolidation of multi-tile, multiyear datasets into large output artifacts whenever reprocessing is triggered by new data or methodological updates.

Overall, traditional reference processing workflows provide a reliable foundation for batch-oriented and event-based analyses but exhibit structural limitations when applied to long EO time series, multisensor archives, and recurrent monitoring scenarios. The absence of mechanisms for incremental updates, baseline reuse, and persistent spatiotemporal representations motivates the exploration of alternative processing architectures, which are examined in the following sections.

2.3 Processing Architectures for Large-Scale Earth Observation Data

The multidimensional and multisensor structure of EO data, formally introduced in Section 2.1.3 (Equations 2.1–2.6), imposes computational demands that grow rapidly with spatial extent, temporal depth and spectral dimensionality. When EO processing workflows are applied repeatedly over long time series and large geographic regions, overall system performance becomes dominated not only by algorithmic complexity, but by architectural factors such as execution orchestration, data locality, and disk I/O.

Let a multisensor EO archive be described by the collection defined in Equations (2.5) – (2.6). For a given sensor s, the cost of applying a generic preprocessing or analysis operator $\mathcal{F}$ over its full temporal archive scales as:

$$C_s \sim \mathcal{O}\left(T_s \cdot H_s \cdot W_s \cdot B_s\right) \tag{2.7}$$

and for a multisensor archive:

$$C_{\text{total}} \sim \mathcal{O}\left(\sum_{s \in S} T_s \cdot H_s \cdot W_s \cdot B_s\right) \tag{2.8}$$

In practice, this theoretical cost is amplified by repeated disk reads and writes, intermediate product generation and redundant recomputation when workflows are executed independently for overlapping spatial or temporal requests. These constraints have driven the development of specialized processing architectures aimed at scaling EO workflows beyond traditional sequential execution.

2.3.1 High-Performance and Distributed Processing Pipelines

High-performance computing (HPC) and distributed batch-processing pipelines represent one of the earliest architectural responses to large-scale EO workloads. In these systems, EO processing chains are decomposed into independent tasks, typically at the scene or tile level, and executed concurrently across multiple compute nodes or CPU cores, a pattern widely adopted in large-scale remote sensing analysis pipelines supporting data-intensive machine learning workflows (Adegun et al., 2023; Burgueño et al., 2023; Guo et al., 2022; Xie et al., 2024).

Examples such as CyberGIS-Compute further demonstrate the evolution of reproducible workflow support on top of traditional HPC EO pipelines (Michels et al., 2024). Similarly, large-scale EO processing systems have achieved substantial efficiency gains through distributed and highly optimized execution, as illustrated by global Sentinel-1 pipelines designed to maximize throughput while remaining fundamentally execution-centric (Agram et al., 2022).

Formally, let $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ denote a set of processing tasks, where each task τ_i applies a fixed operator $\mathcal{F}$ to an independent spatial or temporal subset of the data. The total execution time can be approximated as:

$$T_{\text{exec}} \approx \max_{k \in \mathcal{N}} \sum_{\tau_i \in \mathcal{T}_k} t(\tau_i) \qquad (2.9)$$

where $\mathcal{N}$ denotes the set of compute nodes and $\mathcal{T}_k$ the tasks assigned to node k.

Figure 2.10 schematically illustrates this execution-centric paradigm, where parallelism is achieved by distributing independent scene-level tasks across compute nodes while relying

on disk-based intermediate outputs.

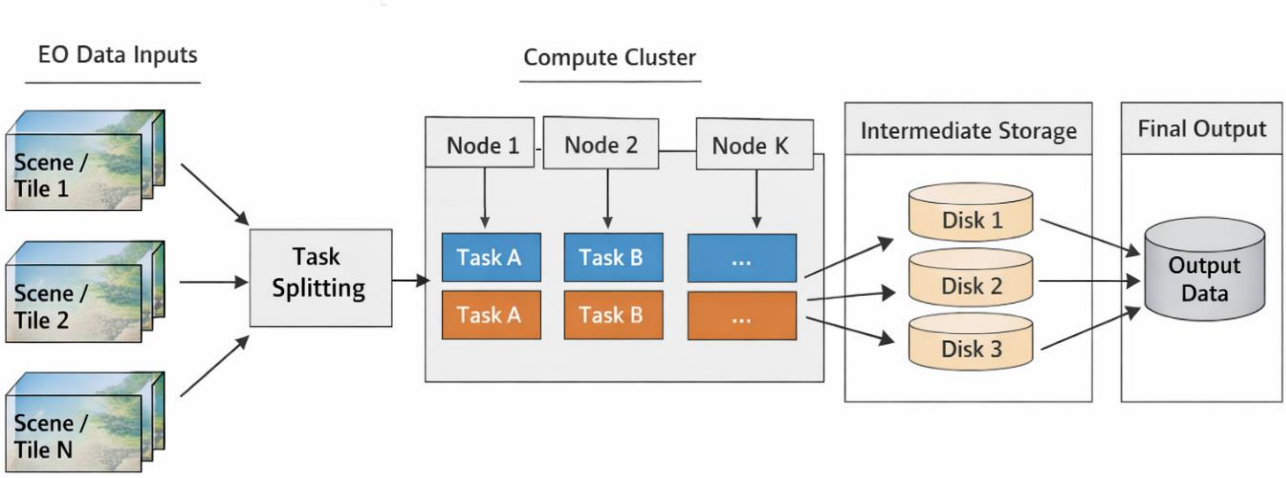


Figure 2.10 Distributed batch-processing pipeline architecture.
Independent scene- or tile-based EO inputs are processed in parallel across multiple compute nodes, with intermediate results written to disk between stages. The model emphasizes task-level parallelism but operates in a stateless manner, leading to redundant computation in recurrent workflows.

This strategy effectively reduces wall-clock time by exploiting task level parallelism and is widely used in large reprocessing campaigns. However, these pipelines remain execution centric: each workflow invocation is treated as an independent job, a design also adopted by recent on-demand EO processing frameworks (Iacono et al., 2023) and intermediate results are materialized on disk between processing stages. Consequently, recurrent analyses over the same areas trigger full re-execution, yielding computational costs that scale linearly with the number of workflow invocations rather than with genuinely new data.

Temporal baselines and historical consistency are not intrinsic to the architecture; they are typically handled through external conventions or manual bookkeeping. As a result, while HPC pipelines improve throughput, they do not address redundancy in long-term or recurrent monitoring scenarios.

2.3.2 Workflow-Oriented and Cloud-Native Architectures

Cloud-native EO architectures extend distributed processing by incorporating elastic resource allocation, containerized execution and workflow orchestration systems, which play a central role in scalable geospatial AI and data-intensive pipelines (Sun et al., 2023). Processing pipelines are typically represented as directed acyclic graphs (DAGs), where nodes correspond to processing operators and edges encode data dependencies.

Let $\mathcal{G} = (\mathcal{V}, \boldsymbol{\mathcal{E}})$ be a DAG representing an EO workflow, where each vertex $v_i \in \mathcal{V}$ corresponds to an operator $\boldsymbol{\mathcal{F}}_i$. The execution cost of the workflow can be expressed as:

$$T_G = \sum_{v_i \in \text{critical path}(G)} t(\mathcal{F}_i), \qquad (2.10)$$

with parallel branches executed concurrently subject to resource constraints.

These architectures significantly improve scalability and fault tolerance and are well suited for large, heterogeneous EO workloads. Cloud-optimized storage and lazy evaluation further reduce unnecessary data transfers by enabling partial reads of large datasets. In practice, most workflow-oriented and cloud-native EO architectures adopt an execution-centric operational model, in which processing is organized around stateless or weakly stateful workflow executions rather than persistent data representations (Gaigalas et al., 2019).

Figure 2.11 illustrates a representative execution-centric EO architecture, where each newly ingested observation triggers an independent workflow execution expressed as a DAG (Deelman et al., 2015). Processing nodes operate as stateless containers orchestrated on demand, while intermediate and final products are written to object storage without maintaining a persistent computational baseline. As a result, historical data are not structurally coupled to newly ingested observations, and repeated requests over overlapping spatio-temporal domains induce redundant execution costs.

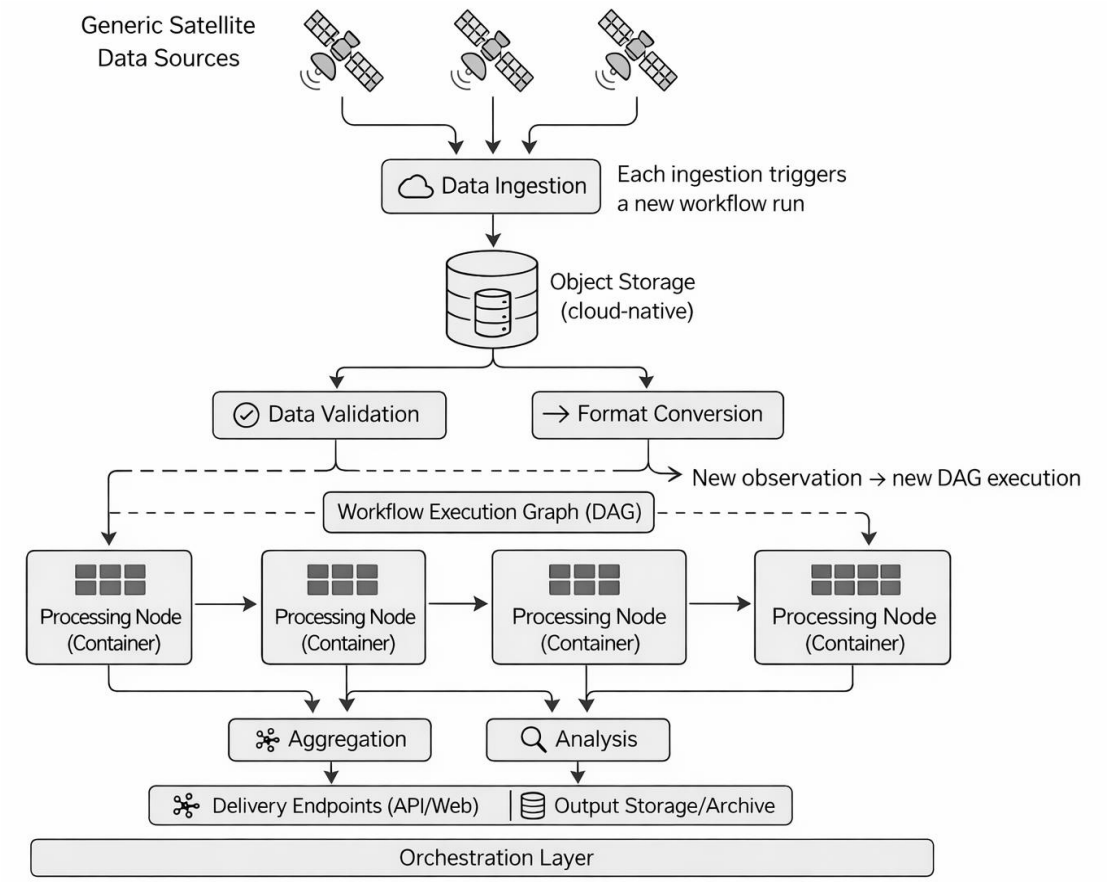


Figure 2.11 Representative execution-centric EO processing architecture.
Each data ingestion event triggers an independent DAG-based workflow executed by stateless containerized nodes, with results written to object storage. The lack of persistent computational state implies that overlapping spatio-temporal requests lead to repeated execution and redundant processing costs.

Nevertheless, the fundamental abstraction remains the workflow execution graph, not the evolving EO dataset. New observations entering the archive typically trigger new DAG executions, even when large fractions of the requested spatio-temporal domain have already been processed. Incremental updates, temporal baseline reuse and historical state management are not first-class architectural concepts, but rather responsibilities delegated to workflow design or user configuration.

2.3.3 Structural Limitations of Execution-Centric Architectures

Despite differences in implementation, both HPC-based and cloud-native EO architectures share a reliance on stateless or weakly stateful execution models. Let $\mathbf{R_j}$ denote a processing request over spatial region $\mathbf{\Omega_j}$ and temporal window $\mathbf{\Delta t_j}$. In execution-centric systems, the cost of handling multiple requests satisfies:

$$C(R_1, \ldots, R_n) \approx \sum_{j=1}^{n} C(R_j), \qquad (2.11)$$

even when:

$$\Omega_i \cap \Omega_j \neq \emptyset \quad \text{and} \quad \Delta t_i \cap \Delta t_j \neq \emptyset.$$

That is, overlapping requests do not reduce computational cost, because previously computed results are not systematically reused.

This behavior becomes particularly inefficient in recurrent monitoring scenarios, where fixed AOI are queried repeatedly as new acquisitions become available. In such contexts, the absence of persistent spatio-temporal state leads to unnecessary recomputation, inflated I/O overhead and limited scalability, even when highly parallel execution and cloud-native storage are employed.

Recent literature emphasizes that addressing these limitations requires architectures capable of reasoning explicitly about processed versus unprocessed data and about temporal continuity within long EO time series (Mahecha et al., 2020; Sun et al., 2023).

2.3.4 Transition Toward Persistent and State-Aware Architectures

The limitations outlined above motivate a shift from execution-centric EO architectures toward state-aware processing paradigms, in which spatio-temporal data structures persist across workflow invocations and evolve incrementally as new observations arrive.

From an architectural perspective, this implies minimizing the marginal cost of incorporating new data. Ideally, the cost of extending an existing archive by $\mathbf{\Delta T}$ new time steps should

satisfy:

$$C_{\text{incremental}} \ll C_{\text{full}}, \qquad (2.12)$$

where C_{full} denotes the cost of reprocessing the entire temporal archive. Achieving this inequality requires persistent representations that decouple historical data from newly ingested observations and allow selective access to spatio-temporal subsets.

This transition establishes the conceptual foundation for multidimensional data cube architectures, which prioritize persistent storage, structured access and reuse over repeated workflow execution. The specific architectural trade-offs introduced by existing data cube implementations are examined in the following section.

2.4 Data Cubes and Multidimensional Architectures

The limitations of execution-centric and workflow-oriented processing architectures have motivated the adoption of data-centric approaches for managing large-scale EO archives. Data cube architectures represent a shift from treating satellite imagery as independent processing inputs toward organizing observations as persistent multidimensional structures indexed along spatial, temporal, and spectral dimensions.

By formalizing EO datasets as structured spatio-temporal arrays, data cubes enable direct subsetting and reuse of previously processed observations, reducing redundant computation and disk I/O. This abstraction emphasizes persistent data organization over repeated workflow execution and provides a coherent framework for accessing long EO time series across multiple spatial extents and sensors.

While data cube architectures address several scalability and reproducibility challenges inherent to traditional processing pipelines, their effectiveness depends strongly on architectural design choices related to data layout, update mechanisms and modularity. These aspects are critical for supporting recurrent monitoring and incremental data growth, particularly in multisensor and long-term EO applications.

2.4.1 Open Data Cube and Monolithic Approaches

Open Data Cube (ODC) architectures represent one of the most influential and widely adopted paradigms for organizing large-scale Earth Observation (EO) archives as multidimensional data structures (Giuliani et al., 2017; Lewis et al., 2017, Cai et al., 2025). In these systems, satellite observations are ingested into a unified spatio-temporal grid, producing a persistent data cube in which spatial, temporal and spectral dimensions are explicitly indexed and standardized. This abstraction has proven highly effective for enabling reproducible analytics, spatial–temporal subsetting, and consistent access to long-term EO time series (Gomes et al., 2020).

At a conceptual level, monolithic data cube architectures can be formalized as a single, persistent multidimensional array:

$$I_{mono} \in \mathbb{R}^{T \times X \times Y \times V}, \qquad (2.13)$$

where T denotes the temporal axis, X and Y the spatial grid, and V the set of stored variables (spectral bands or derived products). All observations contributing to a given spatial, temporal extent are mapped into this common coordinate system during ingestion, and subsequent analyses operate directly on subsets of I_{mono}.

Core Characteristics of Monolithic Data Cube Architectures

Monolithic data cube systems, such as traditional ODC deployments, share a set of defining architectural characteristics:

- **Centralized spatio-temporal grid**: All ingested observations are resampled and aligned onto a single, predefined spatial grid and temporal indexing scheme. This grid acts as the global reference frame for the entire archive.
- **Unified ingestion pipeline**: Data ingestion is performed through standardized workflows that harmonize projection, resolution, and metadata across all products before insertion into the cube.

- **Persistent global baseline:** The cube represents a single, evolving baseline that aggregates all historical observations for a given region or deployment.
- **Query-driven analytics**: User requests are resolved by extracting spatio-temporal subsets from the cube, avoiding repeated scene-level preprocessing.

These properties enable efficient exploratory analysis and reproducible processing over large EO archives and have been successfully applied in national-scale and continental-scale monitoring initiatives (Lewis et al., 2017; Killough, 2018, Wu et al., 2024).

Figure *2.12* schematically illustrates this monolithic organization, where all incoming satellite scenes are ingested into a single spatio-temporal cube aligned to a common grid. Spatial and temporal queries are resolved by extracting subsets from this global structure, which acts as a unique and persistent baseline for all analyses.

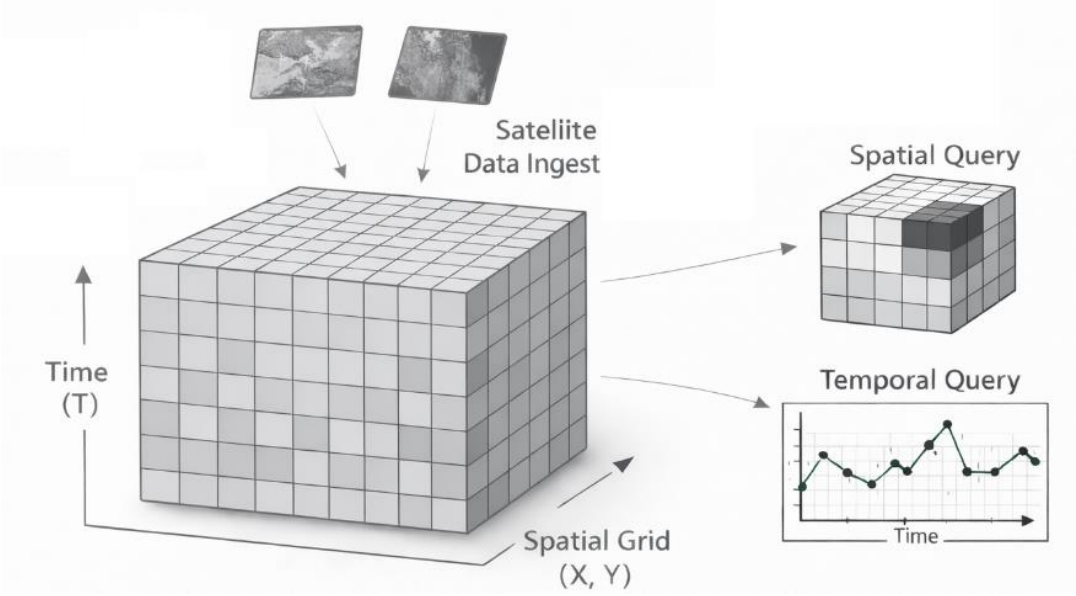


Figure 2.12 Monolithic data cube architecture.
Satellite scenes acquired at different times are ingested into a single, unified spatio-temporal grid, forming a persistent multidimensional cube that serves as a global baseline. All observations are resampled and aligned to the same spatial grid and temporal indexing scheme, enabling query-driven extraction of spatial and temporal subsets from a single authoritative data structure. This architecture emphasizes one cube, one baseline and one global grid.

Advantages of the Monolithic Data Cube Paradigm

Monolithic cube architectures offer several well-established advantages:

- Consistent spatio-temporal representation across the archive
- Reduced redundancy compared to scene-based pipelines
- Reproducible analytics, as all analyses reference the same baseline
- Efficient spatial and temporal slicing, particularly for long-term trend analysis

From a mathematical perspective, repeated analyses over overlapping spatio-temporal domains benefit from reuse of the same underlying data tensor $\boldsymbol{\mathcal{I}}$ avoiding the cost of reconstructing time series from individual scenes (Giuliani et al., 2017).

Structural Limitations of Monolithic Cubes

Despite their strengths, monolithic data cube architectures exhibit structural limitations that become increasingly relevant in recurrent monitoring and incremental-update scenarios (Baumann et al., 2018):

1. **Global coupling of spatial and temporal dimensions** because all observations are ingested into a single global cube, updates affecting any part of the spatial or temporal domain may propagate across the entire structure. In practice, this implies that:
   - Changes in preprocessing assumptions, product definitions, or baselines often require partial or full cube regeneration
   - The cost of updates scales with the size of the cube rather than with the size of the change

2. **Limited support for incremental and recurrent ingestion:** While some ODC implementations support appending new observations, incremental updates are typically constrained by global ingestion rules. New data must conform to the existing grid, resolution, and metadata conventions, and ingestion pipelines are not inherently optimized for frequent, small-scale updates over fixed areas of interest. As a result, recurrent monitoring workflows, where the same spatial regions are revisited continuously, often incur unnecessary ingestion and validation overhead, even when

historical data remain unchanged.

3. **Monolithic baseline as a single point of dependency.** In monolithic architectures, the cube baseline acts as a single authoritative state. This design simplifies global querying but introduces rigidity:
    - All analyses implicitly depend on the same baseline
    - Divergent processing assumptions (alternative corrections or variable definitions) are difficult to isolate
    - Partial reuse of historical data under different assumptions is not natively supported

This contrasts with architectures that allow independent evolution of multiple baselines over different spatial units.

Implications for Recurrent Monitoring Architectures

Monolithic data cube architectures are well suited for centralized, large-area EO analytics where uniform processing assumptions and infrequent baseline changes are acceptable. However, their tightly coupled design poses challenges for scenarios characterized by:

- Recurrent monitoring of fixed areas of interest
- Frequent incremental data arrival
- Heterogeneous temporal coverage across spatial units
- Selective recomputation requirements

In such contexts, the monolithic organization of data and baselines limits architectural flexibility and increases the marginal cost of updates. These limitations motivate alternative designs that preserve the benefits of multidimensional cubes while relaxing global coupling constraints (Baumann et al., 2018; Giuliani et al., 2017*)*, an issue addressed by modular and Multi-cube approaches discussed in subsequent sections

2.4.2 Cloud-Native and Chunked Data Approaches

The limitations of monolithic data cube architectures, particularly their strong global coupling and rigid baseline semantics, have driven the adoption of cloud-native data models designed to improve scalability, flexibility and access efficiency (Sun et al., 2023). Rather than organizing EO archives as a single centralized structure, cloud-native approaches represent data as collections of independently addressable units stored in distributed object storage systems, enabling scalable access across large spatio-temporal domains (Abernathey et al., 2021; Ferreira et al., 2022).

A defining characteristic of these architectures is the use of chunked data layouts, in which multidimensional EO datasets are decomposed into small, fixed-size blocks along spatial, temporal and variable dimensions. Each chunk can be accessed, transferred and processed independently, allowing data access patterns to be driven directly by query extent rather than by the size of the overall archive. This design substantially reduces unnecessary data movement and supports interactive or localized analysis over massive EO collections.

Chunked storage models are tightly coupled with object storage backends, where data blocks are stored as immutable objects and retrieved on demand. Processing frameworks operate directly over these chunks, typically using lazy loading strategies in which only the data required to satisfy a given query are accessed. This decoupling of storage and computation enables efficient parallel execution and elastic scaling across distributed computing environments (Baumann et al., 2018).

Widely adopted cloud-native EO formats such as Zarr, Cloud Optimized GeoTIFF (COG), and chunked NetCDF variants embody these principles, providing optimized access to large EO datasets over standard web protocols without requiring centralized ingestion or monolithic restructuring (Abernathey et al., 2021; Baumann et al., 2021). As a result, cloud-native chunked approaches represent a significant evolution in EO data management, particularly for exploratory analytics and large-scale data dissemination.

Principles of Cloud-Native Chunked Data Models

Cloud-native EO data architectures are characterized by the following principles:

1. **Chunked multidimensional layout**: Data are stored as regularly sized blocks along spatial, temporal, and/or variable dimensions. Chunk shapes are selected to balance input/output efficiency and computational locality.

2. **Object storage as the persistence layer:** Chunks are stored as immutable objects in distributed object storage systems, enabling parallel access and horizontal scalability.

3. **Lazy loading and on-demand access**: Only the chunks intersecting a given query are retrieved, minimizing unnecessary data transfer and memory usage.

4. **Decoupling of storage and computation**: Processing engines operate independently of the physical data layout, accessing chunks remotely as needed.

These principles underpin widely adopted cloud-native EO formats, which are designed for efficient access over HTTP-based object storage (Baumann et al., 2018; Abernathey et al., 2021, Sudmanns et al., 2020) and are schematically illustrated in **Figure 2.13**, where multidimensional EO data are decomposed into independent chunks persisted in object storage. Query execution operates by selectively retrieving only the chunks intersecting the requested spatio-temporal window, while computation remains decoupled from the underlying storage layout.

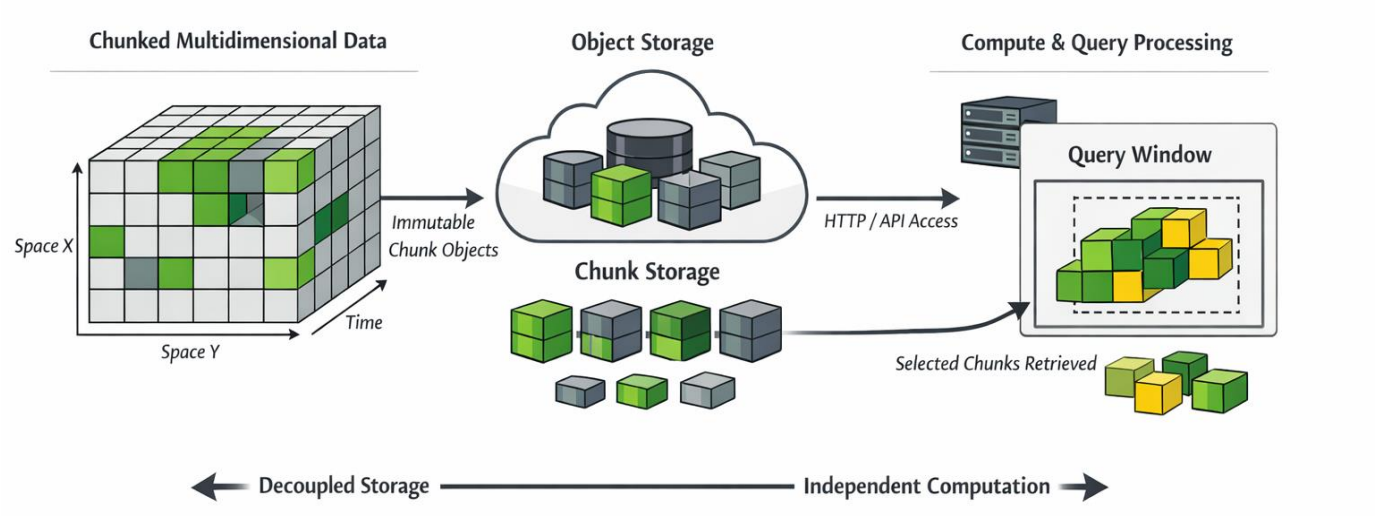


Figure 2.13 Principles of cloud-native chunked data models.
Multidimensional EO data are partitioned into regularly sized, immutable chunks stored as independent objects in distributed object storage. Queries retrieve only the subset of chunks intersecting the requested spatio-temporal window, enabling decoupled storage and computation, efficient parallel access and on-demand processing over large EO archives.

Computational Implications of Chunking

From a computational perspective, chunked data models change the dominant cost drivers of EO analytics by making data access proportional to what is actually requested. Under a chunked layout, a spatio-temporal query triggers retrieval and processing only of those data blocks that intersect the requested time window and spatial footprint. Consequently, both compute time and I/O volume scale primarily with the number of chunks touched by the query, rather than with the total size of the archive. This is a key departure from monolithic cube rebuilds or globally coupled update paths, where large portions of the archive may be scanned or rewritten even when the requested extent is small.

This selective access property supports near-linear scaling with respect to query extent and enables interactive exploration of large EO archives. In addition, because chunks are independent units, they can be scheduled and processed concurrently across distributed compute resources, improving throughput without requiring global coordination beyond query planning and task orchestration (Abernathey et al., 2021; Baumann et al., 2018).

Remaining Limitations of Chunked Cloud-Native Approaches

Despite their advantages, chunked cloud-native data models do not inherently resolve all architectural challenges associated with long-term EO monitoring. In particular:

- Chunking primarily addresses data access efficiency, not baseline management.
- Persistent temporal consistency across repeated analyses is not enforced at the architectural level.
- Incremental ingestion policies and historical state tracking are typically externalized to higher-level systems or user workflows.

As a result, while chunked data formats reduce I/O overhead and improve scalability, they do not by themselves define how evolving EO archives should manage temporal baselines, incremental updates or reuse of previously computed results across monitoring cycles (Abernathey et al., 2021; Mahecha et al., 2020).

2.5 Limitations of Existing Architectures

Despite significant advances in EO data processing infrastructures, current architectural paradigms exhibit structural limitations when applied to large-scale, long-term, and recurrent monitoring scenarios. Although scene-based pipelines, cloud-native workflows, and data cube models address specific aspects of scalability, performance or data access, they do not jointly provide native support for incremental data evolution, baseline reuse and persistent spatio-temporal consistency.

Across the reviewed architectures, a common pattern emerges: processing is either execution-centric or globally coupled, rather than explicitly data-centric. Scene-based and workflow-oriented approaches repeatedly reprocess overlapping spatio-temporal extents, while cloud-native workflows treat each execution as an independent event. Monolithic data cube architectures introduce persistence but enforce a single global baseline, causing update costs and recomputation efforts to scale with archive size rather than with the extent of change (Lewis et al., 2017; Mahecha et al., 2020).

More recent cloud-native and chunked data layouts improve data access efficiency and scalability but primarily operate at the level of storage and I/O optimization. They do not explicitly model temporal baselines, incremental ingestion policies, or recurrent monitoring semantics, leaving these concerns to external orchestration logic or user-defined workflows (Abernathey et al., 2021).

As a result, existing EO architectures lack an integrated mechanism for managing evolving EO archives in which data are continuously appended, partially reused, and selectively recomputed over fixed spatial units. These limitations motivate the need for architectural designs that combine scalable data access with explicit, persistent management of spatio-temporal baselines, enabling efficient recurrent monitoring without global recomputation.

2.6 Need for Incremental and Recurrent Monitoring-Oriented Approaches

Large-scale EO monitoring increasingly relies on repeated analysis of fixed spatial regions as new observations become available over time. In such scenarios, efficiency is determined not by raw processing throughput, but by the ability to incrementally integrate new data, reuse previously processed results, and preserve temporal consistency across successive analyses. The limitations identified in Section 2.5 indicate that existing EO architectures do not natively support these requirements.

An architecture designed for incremental and recurrent monitoring must satisfy the following conditions:

- **Explicit support for incremental ingestion**, enabling new observations to be integrated without triggering global recomputation or full archive rebuilds.
- **Persistent and reusable spatio-temporal baselines**, allowing historical data to remain stable while new data are appended or selectively recomputed.
- **Spatial modularity**, such that processing and updates are confined to well-defined spatial units rather than globally coupled structures.
- **Decoupling of data evolution from execution workflows**, ensuring that repeated

analyses over the same regions do not require repeated end-to-end processing.

- **Selective recomputation mechanisms**, allowing only affected spatial or temporal segments to be updated when processing assumptions change.
- **Autonomous and policy-driven orchestration**, capable of deciding when to ingest, recompute, or reuse existing data without requiring explicit user intervention.
- **Adaptive task parallelization**, where processing is automatically partitioned and scheduled based on data locality, workload size, and available computational resources.
- **Scalability with monitoring frequency**, so that computational and storage costs grow with the amount of new data, not with the total archive size.

Together, these requirements define a shift from execution-centric and monolithic data representations toward architectures that treat recurrent monitoring and incremental evolution as first-class concerns. Addressing these needs requires rethinking how EO data are organized, updated and reused over time, providing the foundation for the architectural approach introduced in the next chapter.

CHAPTER 3

Methodological proposal

3 Methodological Proposal

This chapter presents the methodological proposal of the thesis, which consists of a computational architecture designed to support incremental, recurrent and large-scale processing of satellite image time series. The proposed approach addresses the structural limitations of traditional execution-centric and file-based workflows by introducing persistent spatiotemporal representations and decision-driven process orchestration.

The methodology is based on a Multi-Cube incremental architecture, in which satellite time series are organized into multiple stable data cubes associated with fixed spatial units. These cubes evolve incrementally over time and are reused across successive analyses, enabling efficient management of growing archives while minimizing redundant computation and unnecessary data movement. Incremental updates and partial recomputation are governed by explicit baseline policies that control how new observations are integrated into existing spatiotemporal structures.

A central aspect of the proposed methodology is the integration of an internal orchestration logic that automatically resolves spatial, temporal, and multisensor complexity. Rather than relying on user-driven workflow definitions, the architecture embeds decision mechanisms that determine when to reuse existing data, when to append new information, and when recomputation is required. This design supports recurrent monitoring scenarios and promotes reproducibility and scalability across large and evolving datasets.

The remainder of this chapter describes the main components of the proposed architecture, including the construction of stable data cubes, the definition of dynamic baseline policies, the orchestration and decision logic, multisensor handling and parallel execution strategies.

3.1 Incremental Multi-Cube Architecture

Data cubes now represent one of the most efficient paradigms for handling satellite-acquired big data. Conceptually, a data cube is a multidimensional array in which spatial, temporal, and spectral dimensions are integrated into a coherent structure that enables rapid querying, scalable analytics, and reproducible workflows (Baumann et al., 2018; Appel et al., 2019).

Unlike traditional file-centric repositories, cubes support direct subsetting and computation across time and space, reducing I/O overhead and avoiding redundant preprocessing. Frameworks such as the ODC (Lewis et al., 2019) illustrate the transition towards cloud-native cube infrastructures that allow efficient exploration of multiyear and multi-sensor archives.

Building upon these advances, the proposed architecture introduces a Multi-Cube framework tailored to large-scale satellite time series. Unlike monolithic cubes, this approach generates independent yet interlinked stable cubes whenever a time series must be subdivided due to tiling boundaries or heterogeneous temporal coverage. A stable cube corresponds to a fixed spatial unit whose extent does not change after creation. It is stored in Zarr for efficient cloud-native access and indexed through a spatial hash, allowing it to evolve only temporally through incremental updates.

Each cube operates as a self-contained and persistent object that is reused over time for recurrent monitoring and analysis rather than being created per query. Cubes remain interlinked through shared metadata and hashes, enabling federated querying while preserving internal consistency. A dynamic baseline strategy binds user-defined update policies to the orchestration logic so that only previously unseen dates are incorporated from new NetCDF exports. Each stable cube maintains a local manifest describing its own spatial footprint, temporal coverage, projection and resolution, avoiding the accumulation of a centralized or global metadata catalog. As a result, metadata growth remains partitioned across independent cubes, while baselines evolve incrementally without full archive regeneration.

User interactions are therefore resolved against existing cubes whenever possible. Requests targeting spatial or temporal subsets already contained within a stable cube are handled through direct access and local subsetting, without triggering cross-cube matching or metadata scans. New cubes are created only when no existing cubes satisfy the requested spatial or temporal extent or when required by explicit update policies. This design ensures bounded retrieval complexity and is particularly well suited to continuous monitoring and recurrent analyses over predefined areas.

The Multi-Cube framework follows the modular architecture illustrated in **Figure 3.1** which represents a continuous workflow from user-defined inputs to analytical outputs. The Inputs stage defines the AOI, temporal range, variables and update policy, while the system queries a STAC catalog to identify all items contributing to the requested time series.

Figure 3.1 illustrates the organization of the framework into interconnected functional layers, ranging from user-defined inputs and data discovery to preprocessing, cube orchestration and analytical outputs. Unlike traditional monolithic data-cube designs, the Multi-Cube architecture adheres to an incremental and modular strategy, in which stable data cubes are created, extended or queried without reprocessing the full archive. Visually, the architecture is represented as a collection of hash-indexed stable sub-cubes that evolve independently over time while remaining linked through shared metadata, enabling scalable access, partial updating and flexible spatiotemporal analysis.

The request is then handled by the Orchestrator, who resolves both spatial and temporal planning. Each AOI is converted into a persistent spatial hash, enabling fast retrieval of an existing stable cube or the creation of a new one when required. Temporal completeness is evaluated by comparing requested dates with those already stored. When an AOI intersects with multiple images acquisitions with heterogeneous spatial footprints, the framework activates a subdivision mechanism that splits the region into coherent sub-areas, preserving temporal consistency and avoiding the mosaicking of scenes acquired on different days. This process results in a precise execution plan specifying which spatial units, dates and update policies apply.

In the Preprocess module, all observations required by the execution plan are standardized. For each date, the tile with the greatest spatial overlap is selected and processed through band reading, reprojection, clipping, and variable computation. The resulting per-date NetCDF files act as reusable intermediate layers, preventing redundant preprocessing and enabling selective reconstruction of temporal subsets.

These standardized layers are then integrated into the Incremental Baseline, which maintains the stable Zarr-based data cubes. Depending on the policy selected, the framework either appends only new dates, fills missing temporal gaps, rebuilds a cube, or performs a

recompute-window operation, the last of which, crucially, is a non-mutating path in which the framework resolves the existing stable cube(s) matching the requested spatial and temporal scope and reads only the required chunks to construct a temporary in-memory sub-dataset. No writes are performed for the Zarr baseline, whose layout and temporal chunking are preserved, enabling fast selective access, efficient parallel I/O and long-term consistency without full cube regeneration.

In the Analysis stage, computations are performed either on an updated stable cube or on a temporary in-memory sub-cube, depending on the selected policy, while results are ultimately materialized in the Outputs stage as raster products, vector layers, metrics, or visualizations. Through this sequence of tightly integrated operations, illustrated in **Figure 3.1**, the Multi-Cube framework provides a scalable and reproducible architecture in which ingestion, preprocessing, baseline maintenance and analysis are coherently linked. This design minimizes redundant processing and enables fast, incremental updating across long-term time series, explicitly supporting recurrent and multi-regional monitoring workflows.

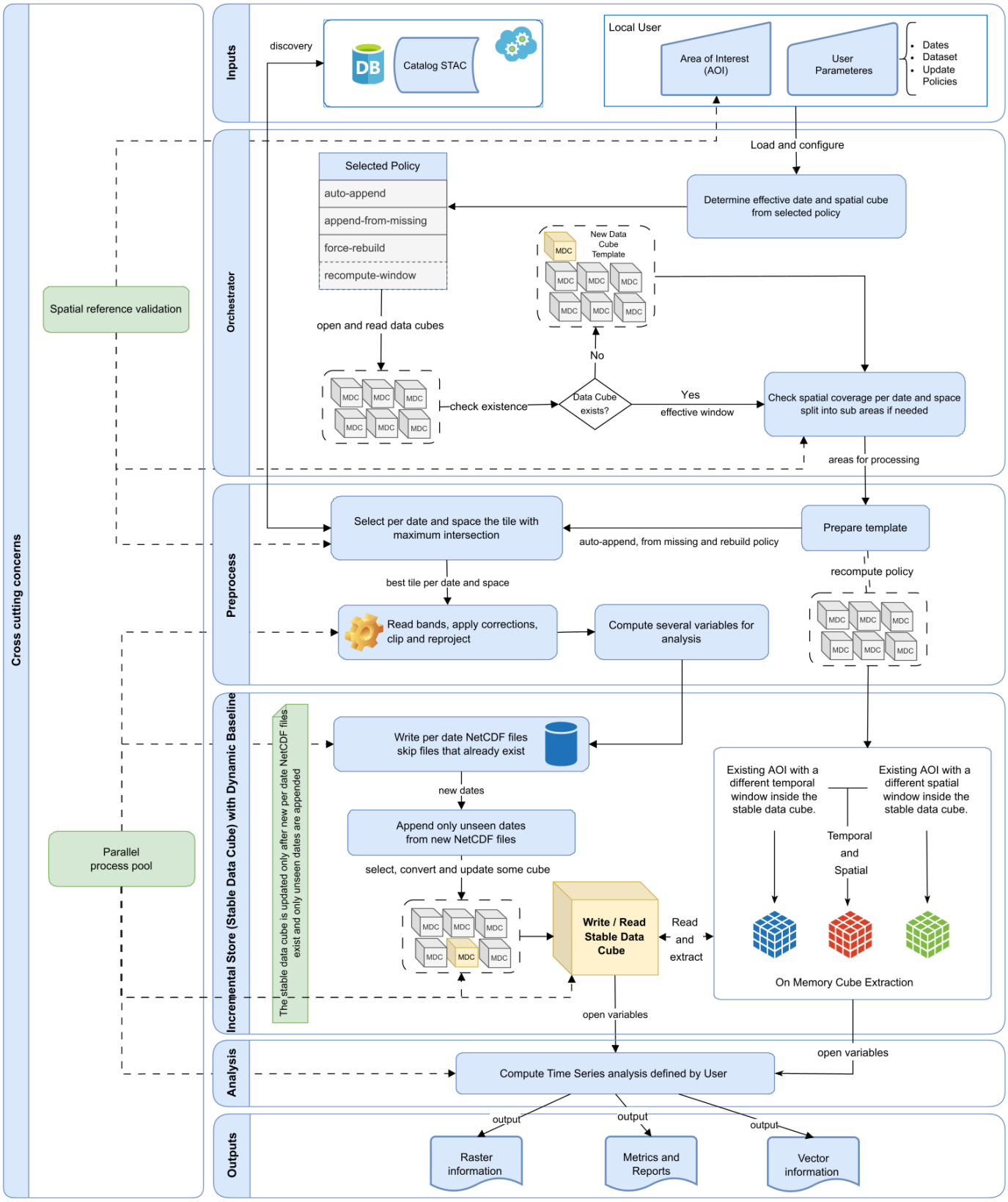


Figure 3.1 Conceptual overview of the proposed Multi-Cube architecture.

From a methodological perspective, the Multi-Cube architecture is conceived not merely as an implementation strategy, but as a generalizable framework for structuring incremental and recurrent EO workflows, independent of specific sensors, study areas or analytical objectives.

3.2 Construcción de Data Cubes estables

The construction of stable data cubes constitutes a core element of the proposed architecture, as it provides the persistent spatiotemporal substrate upon which incremental updates and recurrent analyses are performed. The Multi-Cube framework relies on the Zarr data structure (**Figure 3.2**), a cloud-native format designed for storing and managing large multidimensional arrays. Unlike traditional file-centric formats such as NetCDF or HDF5, which package variables into monolithic files, Zarr organizes data into small, addressable chunks that can be independently stored in local file systems or as objects via cloud storage. This chunk-based organization enables selective access to spatial or temporal subsets without loading entire datasets into memory while supporting efficient parallel I/O through libraries such as xarray and Dask.

Consequently, Zarr significantly improves scalability and performance when handling large satellite archives and has rapidly started being adopted in EO workflows as a foundation for analysis-ready and cloud-optimized data storage.

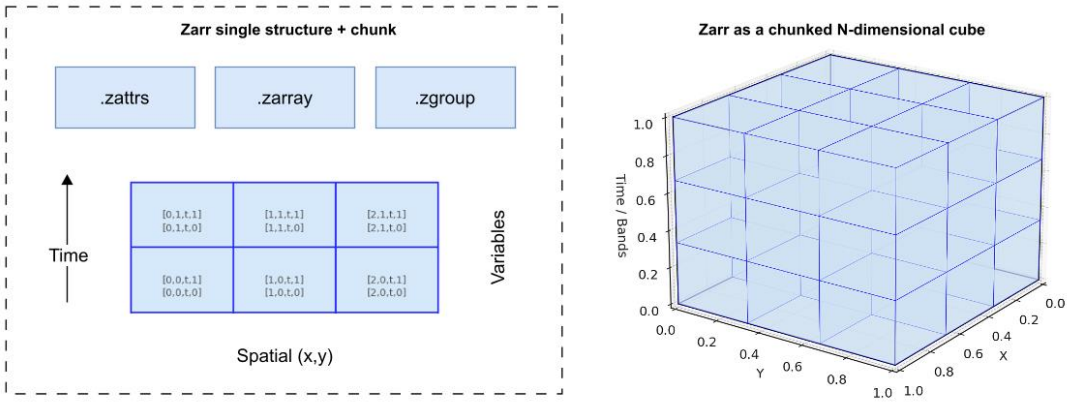


Figure 3.2 Regular Zarr single structure.

The figure illustrates the decomposition of variables into independently addressable chunks along spatial and temporal dimensions, together with the associated metadata files (zattrs, .zarray, .zgroup) that enable scalable and parallel access.

Before applying any baseline update policy, the framework first identifies which stable cube should receive the new observations. This task is achieved through spatial hashing, a mechanism that converts each AOI or sub-geometry into a deterministic unique identifier, ensuring consistent mapping between geometries and their corresponding cubes. As shown in **Equation 3.1** the AOI is serialized into Well-Known Binary (WKB), a standard format for encoding vector geometries defined by the Open Geospatial Consortium (OGC, OGC Zarr Storage Specification, 2022) Simple Feature Access specification, and then hashed using a cryptographic digest. The resulting truncated hexadecimal prefix is used as an area key, allowing direct access to the corresponding cube directory without scanning global catalogs. This guarantees reproducibility and fast lookups even when managing thousands of AOIs.

This design is explicitly aligned with the primary objective of the Multi-Cube framework: efficient recurrent monitoring over fixed AOIs. By associating each recurrent AOI with a stable hash-based identifier, the framework enables rapid and deterministic access to previously constructed cubes, minimizing overhead when the same regions are queried repeatedly as new observations become available.

While the current implementation relies on exact geometry hashing, the same principle can be extended to alternative spatial indexing strategies. For example, grid-based deployments can employ space-filling curve encodings such as Morton (Z-order) indexing, which quantize longitude, latitude and (optionally) time or spectral dimensions into a single location-aware key. Both approaches ensure that incremental ingestion consistently targets the correct baseline cube while preserving flexibility for future extensions.

Formally, for exact geometry hashing, as adopted in the Multi-Cube framework, the following definition is used:

$$H = \mathrm{SHA1}(\mathrm{WKB}(\mathcal{G})), \qquad \mathrm{area\_id} = \mathrm{prefix}_L(H_{\mathrm{hex}}) \tag{3.1}$$

Here, SHA1 is a cryptographic hash function that converts any input into a fixed 160-bit hexadecimal digest, $\mathcal{G}$ is the AOI geometry, H is the digest, and prefix $_L$ extracts the first L hexadecimal characters to create a unique key.

3.3 Dynamic Baseline Policies

A central contribution of this framework is the implementation of a dynamic baseline strategy that governs how new observations are integrated into existing time series archives with different dimensions. In current EO systems, update strategies vary considerably. For example, some STAC-enabled cloud-native repositories and data platforms allow indexing and appending of new scenes without rebuilding the entire archive. In other cases, especially when processing algorithms or baseline versions change, operational collections are typically reprocessed to ensure temporal consistency and cross-sensor comparability.

In this context, the Multi-Cube framework formalizes explicit update policies and binds them to an incremental orchestration logic that minimizes unnecessary recomputation while preserving temporal continuity and reducing computational and storage costs.

In the Multi-Cube approach, the dynamic baseline evolution is controlled by four policies:

- **auto-append,** which extends the cube by appending only unseen dates after the last stored timestamp.

- **append-from-missing**, which reprocesses only those dates within the defined interval that were not successfully ingested, thereby filling temporal gaps.

- **recompute-window**, which isolates a specific temporal or spatial window and reconstructs it independently in memory, without affecting the stable baseline.

- **force-rebuild**, which creates or regenerates the entire cube when required.

This policy-driven design is explicitly aligned with the primary objective of the framework: ensuring efficient recurrent monitoring over stable and repeatedly queried AOIs. In such scenarios, the same AOIs are revisited continuously as new satellite acquisitions become available and rapid access to their historical baselines is essential. By combining deterministic spatial hashing with flexible baseline-updating policies, the framework enables direct and constant access to these recurrent AOIs while ensuring that only strictly necessary recomputation is performed. As a result, long-term monitoring workflows can evolve

incrementally, preserving historical consistency and minimizing both computational overhead and data movement.

Figure 3.3 illustrates the internal Multi-Cube construction workflow, in which each new ingestion request is handled through a policy-driven sequence of coordinated stages. After input validation and stable cube identification via spatial hashing, the framework computes the effective temporal window required by the request and resolves the corresponding update policy. Depending on this policy, the engine either appends new per-date NetCDF layers to an existing baseline, selectively rebuilds specific temporal segments, or performs a read-only recompute-window operation that assembles only the required spatiotemporal slices in memory without modifying the stored cube.

Execution then proceeds by loading and normalizing the necessary files, applying parallel processing through the concurrency manager and writing or assembling the resulting dataset according to the selected policy branch. When baseline mutations occur, metadata are consolidated, and the cube manifest is updated accordingly, whereas non-mutating operations produce temporary in-memory sub-cubes for on-demand analysis. This modular workflow enables efficient handling of entire AOIs or spatially subdivided tiles while preserving temporal integrity and avoiding unnecessary mosaicking and redundant computation.

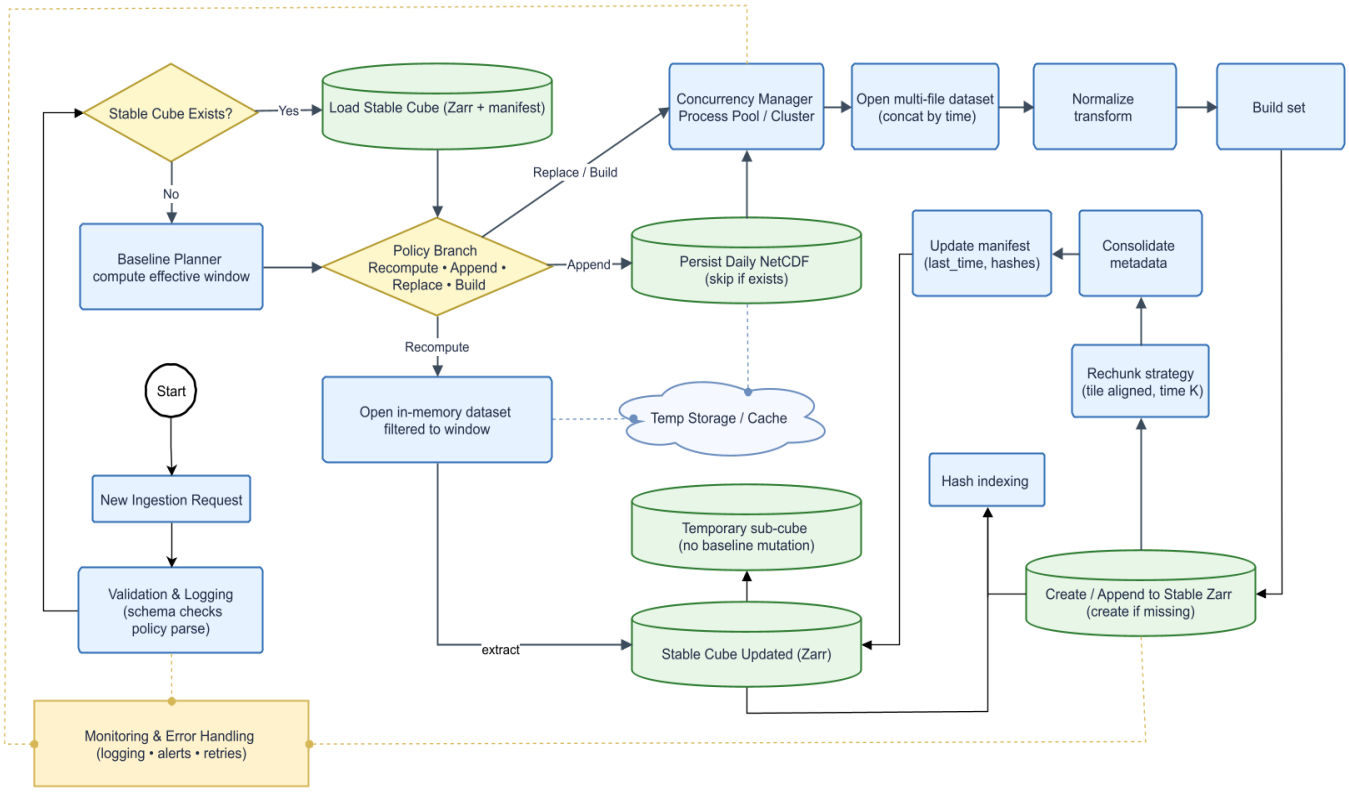


Figure 3.3 Internal workflow of the Multi-Cube incremental engine.

A new ingestion request is resolved through policy-driven decision paths. The diagram shows the sequence of validation, baseline planning and policy selection that determines whether data will be appended to an existing stable cube, selectively rebuilt or extracted as a temporary in-memory sub-cube. The workflow explicitly distinguishes mutating operations, which update the persistent Zarr-based baseline and its associated manifest, from non-mutating recompute-window operations, which assemble read-only sub-cubes without altering the stored data. Concurrency management, validation, monitoring and metadata updates are conducted across multiple branches of the workflow, ensuring consistent and efficient handling of incremental updates and on-demand analyses.

A central aspect of the framework is that the dynamic baseline manages not only temporal updates but also spatial subdivision. When an AOI intersects multiple tiles across different dates, the system automatically generates new hashed cubes, allowing each baseline to evolve modularly and expanding the cube library only when required. This design preserves spatial consistency and temporal integrity while avoiding the systematic mosaicking typical of monolithic cube implementations (Giuliani et al., 2019). By combining incremental ingestion

with Multi-Cube construction, each stable cube, indexed by its spatial footprint, can evolve independently while remaining accessible within a federated archive. This modular organization enhances reproducibility and scalability, addressing challenges also highlighted in regard to Earth System Data Cubes (Augustin et al., 2019) and in studies on interoperability across distributed EO cube instances (Witjes et al., 2023).

In contrast, when a request overlaps multiple existing cubes with heterogeneous spatial or temporal coverage, the framework resolves the situation through an ordered, policy-driven process. Temporal containment is evaluated first to determine whether existing cubes can fully satisfy the requested date range. Spatial overlap is then optimized by prioritizing cubes that provide the largest spatial coverage, minimizing unnecessary subdivision. Only when no existing cube satisfies both spatial and temporal constraints does the framework activate controlled subdivision or trigger new ingestion according to the selected update policy. In such cases, existing stable cubes are treated as independent baselines and are not recombined to form a new cube. This design preserves temporal consistency and avoids implicit mosaicking, while allowing joint interpretation across multiple cubes at the analysis or output stage.

Building on this structure, the framework also accommodates the demands of complex algorithms that function based on long and heterogeneous time series. Incremental baseline management and independent cube evolution provide a flexible computational backend that supports a wide range of analytical workflows without requiring full cube reconstruction. This approach aligns with recent developments in semantic EO data cubes, where scalable structures and enriched metadata enable more advanced and context-aware interpretations (Witjes et al., 2023). The following section illustrates the practical implications of this design by showing how different baseline configurations and update policies lead to distinct performance behaviors when applied to large-scale remote sensing time series.

3.4 Orchestrator and Decision Logic

Rather than being a mere multidimensional repository, the proposed Multi-Cube framework is designed as a decision-driven methodology for incremental EO workflows under heterogeneous usage scenarios. Its contribution lies in an explicit reasoning layer that is intrinsic to the framework and automatically governs how data are organized, subdivided, updated and executed during runtime, without requiring users to manually define spatial partitioning strategies, baseline management rules or execution parameters. In this sense, Multi-Cube emphasizes operational intelligence, how and when to construct, reuse and recompute beyond the underlying storage format.

To situate the contribution of the framework within the existing ecosystem, it is useful to contrast its design philosophy with that of established data cube systems. General-purpose open-source cubes such as ODC (Lewis et al., 2017), gdalcubes (Appel et al., 2019) and other cloud-native implementations primarily address the challenges of data standardization, centralized cataloguing and unified access to long-term EO archives. Their core value lies in transforming disparate satellite products into analysis-ready, multidimensional arrays that can be queried spatially and temporally through standardized APIs, making large archives FAIR.

In contrast, the Multi-Cube framework was designed to be an operational methodology that embeds decision logic to support dynamic and recurrent monitoring workflows, where continuous data arrival and repeated requests over similar regions make “how to update and reuse” as important as “how to store and access”. In this respect, decision-making in Multi-Cube is an integral part of the core architecture. This decision layer governs, for example, when automatic spatial subdivision into independent cubes is required and how execution parameters are adapted to the user’s hardware environment. In comparison, in ODC, such choices are typically delegated to user-defined ingestion pipelines, configuration files, or deployment-specific conventions rather than being resolved automatically by the framework itself.

The Multi-Cube framework therefore operates at a distinct architectural level. Rather than prioritizing the storage or uniform querying of monolithic data structures, it formalizes a

policy-driven orchestration layer that governs how incremental EO workflows should evolve over time. By automating operational decisions that are commonly externalized to manual workflow design or deployment-specific configurations, Multi-Cube provides a flexible and reusable methodological backbone for recurrent monitoring scenarios.

3.5 Multi-Sensor Integration Strategy

The proposed Multi-Cube framework is designed to support multi-sensor EO time series by enforcing a harmonized spatial and temporal representation across heterogeneous data sources. Rather than relying on sensor-specific processing chains or ad hoc mosaicking strategies, the framework adopts a sensor-agnostic architectural approach in which all observations are transformed into a common spatiotemporal reference before being integrated into stable data cubes. This strategy ensures internal consistency, reproducibility and extensibility as additional sensors or collections become available.

Multi-sensor integration is achieved through three tightly coupled components: (i) a sensor-agnostic design layer, (ii) the definition of a master spatial grid and (iii) controlled resampling and harmonization policies that preserve variable semantics while enabling incremental ingestion.

3.5.1 Sensor-Agnostic Architectural Design

At the architectural level, the Multi-Cube framework does not encode sensor-specific assumptions into the orchestration, baseline management or execution logic. Instead, sensor dependency is confined to the preprocessing stage, where raw observations are standardized into a common intermediate representation prior to baseline integration. Let

$$\mathcal{S} = \{s_1, s_2, \ldots, s_n\} \tag{3.2}$$

denote the set of EO sensors contributing observations to the system, where each sensor *s* is characterized by its native spatial grid, spectral configuration and temporal sampling. For each sensor *s*, observations are represented as spatiotemporal fields:

$$X_s(t, x_s, y_s, b_s) \tag{3.3}$$

where t denotes acquisition time, (x_s, y_s) the native spatial coordinates, and b_s the set of sensor-specific variables or bands.

The framework enforces a strict separation between sensor-specific preprocessing and sensor-independent baseline management. Once observations are transformed into the internal representation defined by the master grid (Section 3.5.2), all subsequent operations, including incremental updates, recompute-window extraction, orchestration and analysis, operate identically regardless of the originating sensor. This design enables the extension of the framework to additional EO missions without modifying the core architecture.

3.5.2 Master Grid Definition and Spatial Harmonization

Central to multisensor integration within the proposed framework is the definition of a master spatial grid, which acts as a common spatial reference for all observations ingested into a given stable cube. This grid provides the geometric foundation that enables consistent stacking, comparison, and analysis of observations acquired at different times and by different sensors.

To define such a grid unambiguously, it is necessary to specify how locations on the Earth's surface are represented within the computational domain. This role is fulfilled by the coordinate reference system (CRS), which establishes the mapping between real-world locations and their numerical representation in space. In geographic coordinate systems, positions are expressed using angular measurements such as latitude and longitude, providing a global and sensor-independent spatial reference based on parallels and meridians (**Figure *3.4***)

In the proposed framework, the master grid is defined in a common geographic CRS to ensure interoperability across heterogeneous data products and areas of interest. Consequently, all observations contributing to a stable cube are expressed in a shared latitude–longitude reference frame, and pixel spacing is represented in angular units. When a target spatial resolution is specified in linear units, it is implemented through an equivalent angular sampling that approximates the desired ground spacing at the latitude of the area of interest.

By enforcing a shared CRS at the cube level, the framework guarantees spatial co-registration

of all observations prior to integration, ensuring consistent pixel alignment across time and sensors while preserving a uniform spatial reference for cube construction and subsequent analysis.

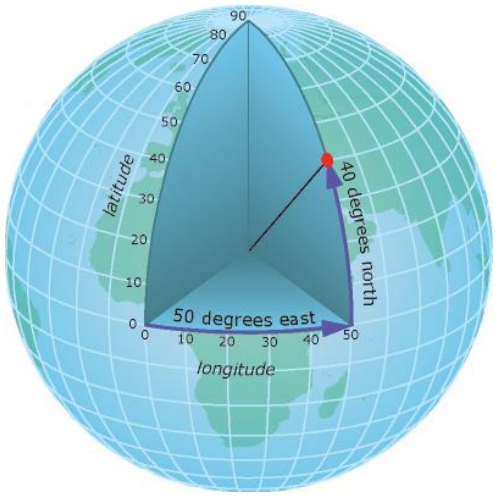


Figure 3.4 Geographic coordinate reference system based on latitude and longitude
Spatial locations on the Earth's surface are represented using angular coordinates defined by parallels and meridians relative to the equator and the prime meridian. This global reference frame provides the basis for expressing all observations in a shared latitude–longitude system prior to their integration into the master grid. Source: ArcGIS Resource Center

In this context, central to multi-sensor integration is the definition of a **master spatial grid**, which acts as a common reference system for all observations ingested into a given stable cube. Let

$$\mathcal{M} = (CRS, \Delta x, \Delta y, \Omega) \tag{3.4}$$

denote the master grid, where:

- *CRS* is the coordinate reference system,
- Δx, Δy are the target spatial resolutions,
- $\Omega \subset \mathbb{R}^2$ defines the spatial extent associated with a stable cube.

The master grid is fixed at cube creation time and remains immutable throughout the cube's lifetime. In practice, the grid resolution and projection are selected based on a reference

sensor or analysis requirement, ensuring compatibility across all contributing observations.

Each sensor $s \in S$ provides data on its native grid

$$\mathcal{M}_s = (CRS_s, \Delta x_s, \Delta y_s, \Omega_s) \tag{3.5}$$

which generally differs from $\boldsymbol{\mathcal{M}}$. To ensure spatial consistency, all observations are mapped onto the master grid via a reprojection–resampling operator $\boldsymbol{\mathcal{R}}_s$

$$X_s^{\mathcal{M}}(t, x, y, b) = \mathcal{R}_s\left(X_s(t, x_s, y_s, b_s)\right) \tag{3.6}$$

This transformation guarantees that all sensor contributions share identical spatial coordinates and can be directly stacked along the temporal and variable dimensions within the stable cube.

3.5.3 Resampling Policies and Variable Semantics

Resampling operations are selected according to the semantic nature of each variable to avoid introducing artifacts or invalid transformations. The framework distinguishes between continuous and categorical variables and applies appropriate resampling strategies accordingly.

Figure *3.5* illustrates the spatial harmonization process applied during multi-sensor preprocessing. Incoming observations defined on heterogeneous native grids are first reprojected to a common coordinate reference system and subsequently resampled onto the master grid according to variable semantics. This distinction between interpolation strategies is central to preserving the physical meaning of continuous variables while maintaining the integrity of categorical or quality-control layers prior to cube integration.

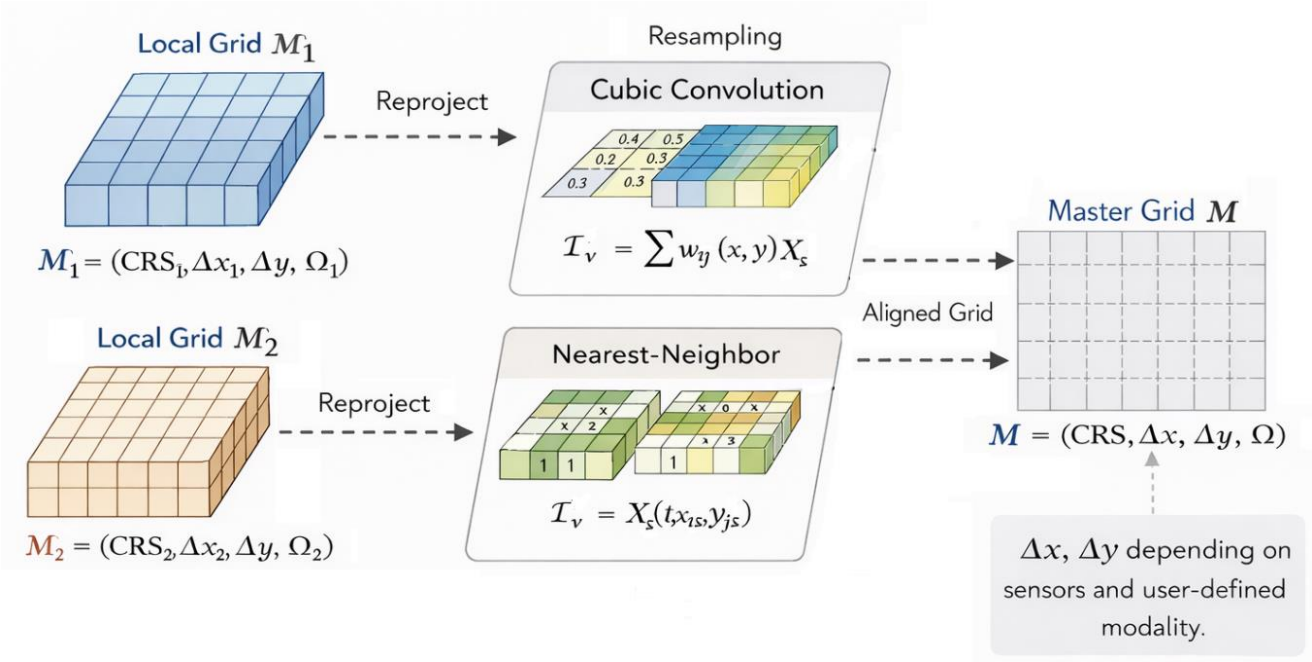


Figure 3.5 Multi-sensor harmonization via reprojection and resampling

Observations from heterogeneous sensors, defined on native grids with different spatial resolutions, are reprojected to a common coordinate reference system and resampled according to variable semantics. Continuous variables are interpolated using cubic convolution, while categorical variables are resampled using nearest-neighbor assignment. The resulting aligned observations are mapped onto the master grid $\mathcal{M}$ whose resolution and extent are defined by the selected integration mode prior to cube integration.

Let v denote a variable associated with a sensor observation. The resampling operator $\mathcal{R}_s$ is decomposed into:

$$\mathcal{R}_s = \mathcal{P}_s \circ \mathcal{I}_v \tag{3.7}$$

where $\mathcal{P}_s$ performs geometric reprojection, and I_v applies variable-specific interpolation.

For continuous variables (reflectance, temperature, spectral indices), interpolation operators such as bilinear or cubic convolution are used:

$$\mathcal{I}_v^{cont}(x,y) = \sum_{i,j} w_{ij}(x,y)\, X_s(t, x_i, y_j) \tag{3.8}$$

For categorical or quality-control variables (classification masks, flags), nearest-neighbor resampling is enforced:

$$\mathcal{I}_v^{cat}(x,y) = X_s(t, x_{i^*}, y_{j^*}) \tag{3.9}$$

where (i^*, j^*) denotes the closest native pixel.

This distinction preserves categorical integrity while ensuring physically meaningful transformations of continuous fields. All resampling decisions are encoded as part of the preprocessing configuration and remain consistent across incremental updates.

3.5.4 Temporal Alignment and Incremental Integration

Temporal harmonization is addressed through explicit handling of heterogeneous revisit frequencies and acquisition schedules. Each observation is associated with its acquisition timestamp t, and no temporal interpolation is performed during ingestion.

The master cube maintains a discrete time axis:

$$\mathcal{T} = \{t_1, t_2, \ldots, t_k\} \tag{3.10}$$

onto which all resampled observations are appended according to the selected baseline policy (Section 3.3). When multiple sensors provide observations on the same date, variables are stored as independent channels or grouped according to analysis requirements, without implicit fusion at ingestion time.

Incremental ingestion proceeds by appending only unseen timestamps or reconstructing bounded temporal windows, ensuring that multi-sensor time series can grow consistently without triggering full baseline regeneration.

3.5.5 Implications for Incremental and Recurrent Monitoring

The combination of a fixed master grid, sensor-agnostic orchestration, and policy-driven baseline updates enables recurrent monitoring workflows over long temporal horizons. Each stable cube evolves independently as new observations become available, regardless of

sensor origin, while preserving spatial and temporal consistency.

By avoiding implicit mosaicking and enforcing deterministic spatial harmonization, the framework supports joint interpretation across multiple sensors at the analysis stage, rather than during ingestion. This design minimizes data duplication, limits unnecessary recomputation, and ensures that historical baselines remain stable as new data streams are incorporated.

3.6 Adaptive Parallelization and Execution

The Multi-Cube framework integrates cloud-native access and parallel execution to address the computational demands of multiyear and multi-tile satellite time series. Given the spatial resolution and acquisition frequency of modern EO data, distributed processing is required to outperform traditional sequential pipelines. Accordingly, the framework employs Dask-based parallelization as an execution backend, enabling concurrent execution of band reading, reprojection, clipping and variable computation across spatial subdivisions defined by the orchestrator while maintaining stable memory usage.

Cloud infrastructures further enhance scalability by providing STAC-enabled, analysis-ready assets, enabling on-demand access to optimized images data while reducing I/O overhead by avoiding full local downloads. Recent advances in cloud-native EO ecosystems show that hybrid on-premise and cloud processing can sustain long-term analytical workloads (Lawler et al., 2025) while access to distributed holdings across providers improves operational flexibility and data throughput (van der Meer et al., 2022). Within this context, the framework leverages chunked processing and lazy evaluation to ensure that only relevant portions of the archive are accessed, further improving efficiency (Munteanu et al., 2024).

In addition, the framework incorporates an adaptive execution layer that automatically inspects the operating system and available computational resources during runtime. Based on this information, the orchestrator dynamically configures process pools and concurrency levels across CPU cores, allowing parallel execution to be adjusted to the characteristics of the host environment without requiring manual tuning. This adaptive strategy prioritizes stable throughput and controlled resource usage rather than maximal parallelism.

Parallelization and cloud-native access operate jointly with the incremental ingestion strategy of the dynamic baseline. Spatial subdivision is processed independently, daily NetCDF files are generated or skipped depending on the update policy, and only the required temporal slices are appended to the stable Zarr store. This modular form of execution avoids full pipeline regeneration and supports scalable long-term monitoring across multiple regions, aligning the Multi-Cube framework with recent advances in adaptive and semantic EO data cubes (Kempeneers et al., 2022; Kröber et al., 2025).

In summary, the proposed methodology redefines EO data-cube processing by shifting the architectural focus from execution-driven workflows to persistent, policy-governed spatiotemporal structures. By treating incremental evolution and recurrent monitoring as first-class architectural concerns, the Multi-Cube framework establishes a methodological foundation that is subsequently evaluated through concrete implementations and controlled experiments in Chapter 4.

CHAPTER 4

Experimental design and implementation

4 Experimental Design and Implementation

4.1 Considerations of Implementation

The experimental design presented in this chapter builds directly upon the architectural principles introduced in Chapter 3 and reflects a set of concrete implementation decisions required to operationalize the proposed Multi-Cube framework. These decisions are not incidental, but rather play a central role in determining how the framework behaves under realistic processing conditions and how its architectural properties can be empirically evaluated.

The implementation targets large-scale, multiyear EO time series characterized by high spatial resolution, dense temporal sampling and repeated access over fixed AOIs. As a result, the design prioritizes incremental ingestion, selective data access and controlled resource usage over monolithic processing or full archive reconstruction. This emphasis directly informs the choice of data formats, execution model and parallelization strategy adopted in the experiments.

At the storage level, the framework relies on cloud-native, chunked data representations to enable efficient access to localized spatio-temporal subsets. Persistent stable data cubes are maintained as immutable baselines that evolve incrementally through policy-driven updates, while intermediate and temporary products are explicitly separated from long-term storage. This separation allows experimental scenarios to isolate the cost of baseline updates from on-demand analytical operations, which is essential for evaluating the benefits of incremental execution.

From an execution perspective, the implementation follows a hybrid model that supports both local and cloud-based data access. Rather than assuming a fully cloud-hosted environment, the framework is designed to operate across heterogeneous infrastructures, including on-premise systems accessing remote object storage. This choice reflects common operational constraints in research and institutional settings and ensures that experimental results remain representative of real-world usage patterns.

Parallel execution is implemented as an adaptive mechanism rather than a fixed configuration. Concurrency levels and process pools are dynamically adjusted at runtime based on the available computational resources, allowing the same experimental workflow to be executed across different hardware environments without manual tuning. This adaptive behavior is particularly relevant for evaluating the scalability of the framework under varying resource constraints.

Together, these implementation considerations define the experimental boundary conditions under which the framework is evaluated in the following sections. By explicitly documenting the underlying execution and storage assumptions, this section provides the necessary context for interpreting the experimental configuration, use cases, and performance metrics presented throughout the remainder of this chapter.

4.2 Data Sources Used

The experimental evaluation of the proposed Multi-Cube framework is conducted using satellite imagery from the Sentinel-2 mission. Sentinel-2 represents one of the most widely adopted open-access EO data sources and provides a particularly suitable testbed for validating architectures designed to manage large-scale, multiyear time series under recurrent monitoring scenarios.

For the experiments presented in this chapter, Sentinel-2 imagery is used as the primary data source to assess the framework's ability to support incremental ingestion, spatial subdivision, dynamic baseline updates and on-demand extraction of spatio-temporal subsets. The choice of Sentinel-2 is therefore motivated by its capacity to stress the framework along multiple dimensions, spatial resolution, temporal density and data volume, rather than by any sensor specific dependency.

Sentinel-2 is an optical-imaging mission of the European Union's Copernicus EO program providing multispectral 10 m spatial-resolution imagery of the whole planet. Since the launch of Sentinel-2A in 2015, Sentinel-2B in 2017 and Sentinel-2C in 2024, the mission has delivered global coverage every five days at the equator through its MultiSpectral Instrument (MSI) and covers 13 spectral bands (**Table 1**) from visible to shortwave infrared with 10 m,

20 m and 60 m resolutions, enabling detailed mapping of vegetation, water bodies, urban areas and land cover change over seasons and years.

Band name	Resolution (m)	Central wavelength (nm)	Bandwidth (nm)	Description
B01	60	443	20	Coastal Aerosol
B02	10	490	65	Blue
B03	10	560	35	Green
B04	10	665	30	Red
B05	20	705	15	Red Edge
B06	20	740	15	Red Edge
B07	20	783	20	Red Edge
B08	10	842	115	Near Infrared
B08A	20	865	20	Red Edge
B09	60	945	20	Water vapor
B10	60	1375	30	Cirrus
B11	20	1610	90	Short-Wave Infrared
B12	20	2190	180	Short-Wave Infrared

Table 1 Sentinel-2 spectral bands

The mission delivers two key data products: Level-1C products are top of atmosphere reflectance images that are orthorectified and organized into UTM/WGS84 tiles of roughly 100 × 100 km and distributed in the SAFE format with JPEG2000 bands and rich metadata. Subsequently, Level-2A products apply atmospheric correction and generate bottom of atmosphere reflectance alongside a scene classification layer (SCL) using the Sen2Cor processor, enabling cloud, water, vegetation and shadow masking (ESA. Sentinel-2 Level-1C Product Format Specification). These well-engineered data streams make the mission ideal for time series analysis, trend monitoring, and change detection.

Researchers and practitioners worldwide have harnessed Sentinel-2 data across diverse domains. In agricultural monitoring, red-edge metrics and spectral indices are exploited to assess crop phenology, stress and yield (Weiss et al., 2020). Water quality specialists use this dataset's spectral reach to map turbidity and chlorophyll in inland and coastal waters through tailored correction algorithms (Pahlevan et al., 2022). At regional to global scales, analyses

track deforestation, land use change and urban expansion through multitemporal classification models (Li et al., 2018). Emergency response teams apply spectral indices and SCL masking to identify burned areas, floods, and damage after natural disasters (Gaveau et al., 2021).

Overall, Sentinel-2 combines fine spatial detail with frequent revisit cycles, providing a robust observational baseline for Earth monitoring. At the same time, its rich data delivery capacity challenges analysts to adopt efficient, incremental, and cloud-enabled workflows to fully exploit its potential for long-term landscape change analysis.

From an experimental standpoint, Sentinel-2 constitutes the highest spatial resolution satellite mission that is currently available as a globally consistent and freely accessible data source. Its open data policy and standardized processing levels enable reproducibility across different study areas and experimental configurations, which is a key requirement for architectural validation. In addition, the richness of its spectral bands allows the framework to be evaluated under conditions that involve multiple variables with distinct physical and semantic characteristics.

4.3 Experimental Configuration

The experimental configuration defines the conditions under which the proposed Multi-Cube framework is evaluated and establishes the context for the validation scenarios and performance analyses presented in this chapter. The experiments are designed to reflect realistic operational workloads, characterized by long-term satellite time series, recurrent access to fixed areas of interest and incremental data ingestion.

Experiments are conducted over a set of predefined AOIs, which are treated as persistent spatial units and mapped deterministically to stable data cubes through spatial hashing, as described in Chapter 3. This configuration enables the evaluation of baseline reuse, spatial subdivision, and incremental updates when new observations become available.

The temporal scope spans multiple years of satellite observations, allowing the framework to be tested under both complete time-series processing and localized, on-demand access patterns. All data are preprocessed following the harmonization strategy introduced in

Section 3.5, ensuring spatial consistency across observations prior to cube integration.

Execution relies on the adaptive parallelization strategy described in Section 3.6, which dynamically adjusts concurrency levels according to available computational resources. Together, these configuration choices provide a controlled yet representative setting for assessing the architectural behavior and computational performance of the proposed framework in the following sections.

4.3.1 Study Areas

Selected study areas suitable for demonstrating the effectiveness of the Multi-Cube framework in handling multiple polygons of varying extents, each associated with distinct time series and parallel analytical processes. Several polygons were deliberately located at Sentinel-2 tile boundaries, reflecting common operational scenarios where consecutive satellite orbits do not provide identical sensing dates and partial tile overlap occurs.

Some study areas span multiple tiles or require more than one tile for complete coverage. In such cases, the framework activates the subdivision mechanism, splitting the AOI into independent sub-cubes that preserve temporal consistency without mosaicking during ingestion time.

Three study areas with large spatial extents were selected, with Areas A, B and C covering 258 km², 317 km² and 343 km² respectively (**Figure 4.1**). These areas were designed to validate the decision-making capacity of the framework under different spatial configurations. Area A is fully contained within a single Sentinel-2 tile, representing the ideal case in which no spatial subdivision is required. Area B intersects two tiles; however, one tile provides complete coverage, allowing the framework to apply a single-tile policy and avoid unnecessary subdivision. In contrast, Area C overlaps two tiles without full coverage from either, triggering the splitting mechanism that divides the study area into two independent sub-areas, each associated with its own time series cube.

In addition, the input AOI consists of a multi-polygon with spatially independent regions. The framework automatically identifies and resolves each polygon independently during cube construction, generating separate stable cubes without user intervention and applying

the same decision logic used for tile-based subdivision.

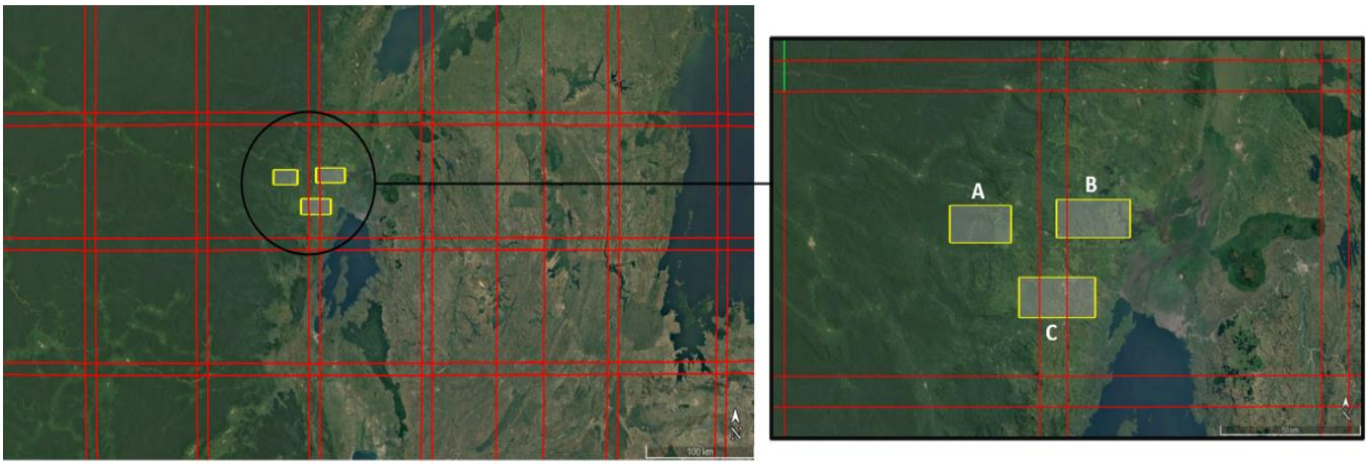


Figure 4.1 Study areas selected to test the framework over a Google Earth basemap.

The areas illustrated in **Figure 4.1** correspond to the original input regions A, B and C provided to the framework. During cube construction, area C is automatically subdivided into two stable cubes (C-1 and C-2) because of incomplete tile coverage, resulting in the four experimental cubes later visualized in the performance analysis.

Table *2* summarizes the main spatial, temporal and data-volume characteristics of each constructed cube used in the experimental evaluation.

Attribute	Cube A	Cube B	Cube C-1	Cube C-2
AOI Coverage (%)	100	100	27	73
Spatial extent WGS84	[28.515, -1.427] [28.709, -1.319]	[28.852, -1.413] [29.085, -1.303]	[28.734, -1.640] [28.798, -1.525]	[28.797, -1.640] [28.974, -1.525]
Dimensions (X × Y)	2153 × 1190	2589 × 1218	710 × 1276	1962 × 1276
Area (km²)	258	317	91	252
Sentinel-2 images	598	598	599	598
Total image data	20930	20930	20965	20930

Table 2 Characteristics of the stable cubes generated for the experimental evaluation

4.3.2 Computational Environment

All experiments were executed on a dedicated workstation in order to ensure a stable and reproducible computational environment. The system configuration was selected to reflect a realistic research-oriented setup rather than a specialized high-performance computing infrastructure, allowing the observed execution behavior to be interpreted in an operational context.

The workstation provides a multi-core CPU architecture and sufficient memory to support parallel processing of multi-temporal satellite datasets, while local storage is used for intermediate products and persistent data cubes. Remote access to cloud-hosted EO data is performed through STAC-compliant services, enabling on-demand retrieval of analysis-ready assets without requiring full local downloads.

The main hardware and software characteristics of the execution environment are summarized in **Table 3**.

Component	**Specification**
Operating System	Linux Mint 22 (Ubuntu 24.04 base), Kernel 6.8.0-64, x86_64 architecture
CPU	AMD Ryzen 9 5900X, 12 cores / 24 threads, base 2.2 GHz, boost up to 3.7 GHz
RAM	128 GB DDR4
GPU	NVIDIA GeForce GTX 1080, driver 570.169
Storage	NVMe SSD 1 TB + SSD 4 TB + 3 × 10 TB HDD
Total Storage	~31.8 TiB
Motherboard	ASUS PRIME X570-PRO, UEFI BIOS v3801
Network	Intel Gigabit Ethernet (I211 and 82574L controllers)

Table 3 Hardware specifications.

With the experimental configuration and computational environment thus defined, the following section focuses on a set of representative use cases designed to validate the architectural behavior of the proposed Multi-Cube framework. These use cases are not intended to assess specific analytical algorithms, but rather to examine how different execution modes, baseline update policies, and access patterns interact with the proposed architecture under realistic monitoring scenarios.

4.4 Use Cases for Architectural Validation

To validate the architectural behavior of the proposed Multi-Cube framework under different operational conditions, a set of representative use cases is defined. Each use case is designed to stress specific architectural components of the framework, focusing on execution modes, data access patterns, and baseline management strategies rather than on application-level analytical results. The following use cases are considered:

Use Case 1, Sequential versus Adaptive Parallel Execution: This use case evaluates how the framework behaves under different execution configurations, with the objective of validating the adaptive parallelization strategy described in Section 3.6.

- **Sequential baseline mode**: Parallel execution is intentionally disabled to simulate a lower-bound computational scenario. This configuration represents a conservative execution mode comparable to traditional sequential pipelines.
- **Adaptive parallel mode**: The framework autonomously configures concurrency levels at runtime, allocating workers and exploiting multi-threading based on available computational resources. This mode reflects the default operational behavior of the Multi-Cube framework.

This use case is intended to assess how adaptive execution influences pipeline flow, task scheduling, and overall resource utilization without modifying the underlying data organization.

Use Case 2, On-Demand Spatio-Temporal Sub-Cube Access: This use case examines the framework's ability to provide immediate access to localized subsets of existing stable cubes, validating the non-mutating execution paths introduced by the dynamic baseline strategy.

- **On-demand sub-cube extraction**: The framework extracts spatially bounded regions and restricted temporal windows directly from existing stable cubes, assembling the requested data in memory without triggering baseline updates.

- **Rebuild-centric reference workflow**: The same requests are conceptually contrasted

with conventional workflows in which even small-area or short-period queries require full cube reconstruction.

The objective of this use case is to validate selective data access, baseline reuse, and avoidance of unnecessary recomputation when handling localized analytical requests.

Together, these use cases define a structured experimental basis for evaluating the architectural properties of the proposed framework. The quantitative computational performance associated with each scenario is analyzed in section 5.

4.5 Computational Performance Metrics

The computational performance of the proposed Multi-Cube framework was evaluated using a set of quantitative metrics designed to characterize execution efficiency, resource utilization, and scalability under different operational configurations. These metrics were selected to capture both end-to-end behavior and internal execution dynamics, allowing a direct comparison between sequential, rebuild-centric workflows and the proposed incremental, policy-driven architecture.

The evaluation focuses on five complementary metric categories, summarized below.

4.5.1 End-to-End Execution Time

Overall runtime was measured as the total wall-clock time required to complete each experiment, from ingestion request initialization to final output generation.

- **Metric:** Total runtime (hh:mm:ss)
- **Purpose:** Quantify global performance gains achieved by incremental execution and parallelization.
- **Reference configuration:** Sequential emulation with all parallelism disabled.
- **Comparison configuration:** Adaptive Multi-Cube execution with automated concurrency.

This metric captures the cumulative impact of orchestration, data access, preprocessing, baseline reuse and analysis stages.

4.5.2 CPU Utilization and Concurrency

To characterize how computational resources are exploited, multiple CPU-related metrics were recorded during execution:

- Mean CPU utilization (%)
- Peak CPU utilization (%)
- Mean worker concurrency
- Peak worker concurrency

Concurrency is defined as the number of simultaneously active worker processes executing tasks at a given time. These metrics allow evaluation of how effectively the framework distributes work across available cores and sustains parallel execution over long processing intervals.

Worker lifetimes and overlap patterns were also logged to analyze task scheduling behavior and temporal overlap between processing stages.

4.5.3 Memory Usage

Memory consumption was monitored to ensure that performance gains were not achieved at the cost of excessive memory usage.

- **Metric:** Main-process RAM usage
    - 90th percentile (p90)
    - Peak memory usage (GB)

These metrics verify that chunked access, lazy evaluation and incremental ingestion maintain stable memory footprints even under high concurrency.

4.5.4 Disk I/O Volume

Disk input/output operations were measured to quantify reductions in redundant data

movement, a central design goal of the Multi-Cube architecture.

- **Metrics:**
    - Total disk read volume (GB)
    - Total disk write volume (GB)

These measurements are particularly relevant for contrasting rebuild-centric workflows, which rewrite entire cubes, against incremental and on-demand sub-cube access modes, which limit writes to unseen dates or avoid baseline mutation altogether.

4.5.5 Data Volume and Processing Scale

All experiments were conducted using an identical data volume to ensure comparability across configurations:

- **Total processed:** 83,755 images
- **Temporal coverage:** 2016–2024
- **Spatial configuration:** Multiple AOIs with heterogeneous tile coverage and automatic subdivision

This fixed-scale setup ensures that observed performance differences arise from architectural behavior rather than differences in data size or content.

Metric Category	Metrics Evaluated	Objective
Execution time	Total runtime	Global efficiency
CPU usage	Mean & peak CPU, concurrency	Parallel effectiveness
Memory usage	RAM p90 & peak	Stability under load
Disk I/O	Read & write volume	I/O efficiency
Scale	Images processed	Architectural robustness

Table 4 Summary of Performance Metrics

The metrics defined in this section provide a comprehensive basis for evaluating how the proposed Multi-Cube framework improves computational efficiency through incremental ingestion, adaptive parallel execution, and baseline-aware data access. The following section presents the experimental results obtained under these metrics, highlighting the performance differences between sequential, rebuild-centric workflows and the proposed decision-driven architecture.

CHAPTER 5

Results and discussion

5 Results and Discussion

The proposed Multi-Cube framework was evaluated under realistic large-scale EO workloads characterized by long satellite image time series, heterogeneous spatial coverage, and recurrent access to fixed AOIs. The experiments were conducted over a single composite AOI, represented as a MultiPolygon composed of three spatially disjoint polygons located in the central rainforest of the Congo Basin (Figure 4.1) using Sentinel-2 Level-2A imagery acquired between 2016 and 2024. For each AOI, stable Multi-Cubes were constructed to store harmonized, spatially standardized and temporally aligned multivariate time series. During cube construction, multiple categories of inputs harmonized in the preprocessing stage were integrated, including:

- Sentinel-2 Level 2A bands
- Quality and SCL for masking clouds and shadows
- Derived spectral indices
- Metadata attributes standardized from the STAC catalog, including acquisition date, tile ID and quality indicators.

Within this experimental setting, it was integrated the semi-supervised and multitemporal event detection algorithm developed by Deijns et al., 2024. Their method, originally designed for unexplored regions, detects geomorphic hazard impacts such as landslides and flash floods in Sentinel-2 imagery by computing the cumulative difference with respect to the mean curves for multiple spectral indices. Sudden disturbances produce a distinctive triangular signal whose peak indicates the timing of the event. Identifying these peaks requires completing mathematically intensive signal processing steps including optimized peak finding algorithms applied over massive multidimensional arrays supported by operations such as array transposition and stride reduction.

This approach is well suited to large-scale time series analysis in regions for which there are few ground-based observations, making it an appropriate use case for evaluating the Multi-Cube framework. The workflow relies on long, multivariate and temporally aligned series, which directly benefit from the framework's ability to standardize inputs, parallelize preprocessing and manage multiple AOIs with heterogeneous spatial and temporal

characteristics. This enables methods such as Deijns et al., 2024. to be reapplied efficiently over the same stable cubes without modifying the underlying baseline, supporting recurrent monitoring over fixed areas, which is a core objective of the proposed event detection methodology.

In this context, the algorithm is integrated as an external analysis stage that directly consumes the stable Multi-Cubes, reinforcing the generality of the proposed architecture. Any method requiring spatially consistent and temporally aligned variables can be applied to the cube as a standardized analytical interface. This integration exercise demonstrates that the methodology developed by Deijns et al., 2024. integrates naturally within the Multi-Cube environment and that the framework can support multi-AOI monitoring workflows while remaining extensible to additional dimensions in future applications.

5.1 Scenario 1: Sequential Emulation vs. Adaptive Parallelization

Using the machine reported in **Table *3*** and the dataset described in the previous section in **Table *2***, it was executed two end-to-end runs of the Multi-Cube base pipeline. The first run was a controlled sequential baseline in which the same scientific routines as the full pipeline were executed but with all parallelism disabled:

- Execution is restricted to a single CPU core
- Numerical and I/O libraries are limited to one processing thread
- There is no subdivision of the time series into smaller sub-areas (tiling disabled)
- A single worker handles all stages (data loading, transformation, and computation), ensuring a strictly serial process.

The second run leveraged the framework's full parallel potential:

- The framework automatically adjusts the number of workers assigned to each processing stage according to system capacity.
- Time series are divided into smaller independent spatial–temporal blocks that can be processed simultaneously.
- External numerical libraries are constrained to a controlled number of threads to

prevent excessive contention for CPU resources.

- Data are read and processed in spatial and temporal chunks so that I/O and computation can proceed concurrently.

The optimized Multi-Cube framework enables scalable execution by orchestrating small, independent tasks and adapting concurrency to the available hardware, whereas the sequential baseline disables these mechanisms and serves as a clean reference in which performance differences arise solely from orchestration rather than from changes in the analytical code.

Under these conditions, end-to-end runtime was reduced from approximately **52 hours** in the sequential mode to about **9 hours** in the fully optimized Multi-Cube run. System monitoring indicated that the level of CPU usage for sequential execution remained close to that for single-core CPU usage, whereas the Multi-Cube configuration sustained high overall CPU utilization with workers working concurrently. Given comparable I/O volumes, the performance gain mainly reflects an acceleration of CPU-bound stages such as reprojection, resampling, and per-pixel time series analysis enabled by multilayer parallel scheduling. The principal results are summarized in this table:

Metric	Sequential	Parallel	Ratio
Runtime (hh:mm:ss)	52:48:20	09:40:18	5.46×
Mean CPU (%)	9.5	27.3	
Peak CPU (%)	102.6	530.8	
Main-process RAM p90 / peak (GB)	37.42 / 54.53	38.91 / 54.96	
Disk read (GB)	6.58	6.04	
Disk write (GB)	157.90	109.91	
Mean concurrency	4.02	14.23	3.54×
Peak concurrency	10	24	2.40×

Table 5 Main performance summary (sequential vs. parallel)

The sequential configuration showed near-serial execution with minimal concurrency and low multi-core utilization, resulting in extended idle periods. In contrast, the parallel configuration sustained concurrent task execution, achieving greater aggregate CPU usage and much shorter runtimes. These dynamics explain the performance gap observed in **Figure *5.1***.

In the sequential configuration (**Figure *5.1*a**), worker processes appear in short, mostly non-overlapping bursts, leading to extended periods with few active processes and limited concurrency. In contrast, the parallel configuration (**Figure *5.1*b**) exhibits dense temporal overlap and stable plateaus of concurrent workers, with diagonal patterns reflecting the continuous replacement of completed tasks.

Although the dense overlapping segments in the parallel case make individual worker lifetimes harder to distinguish visually, they provide clear evidence of sustained task overlap and efficient handoffs, in contrast to the near-serial execution observed in the sequential scenario.

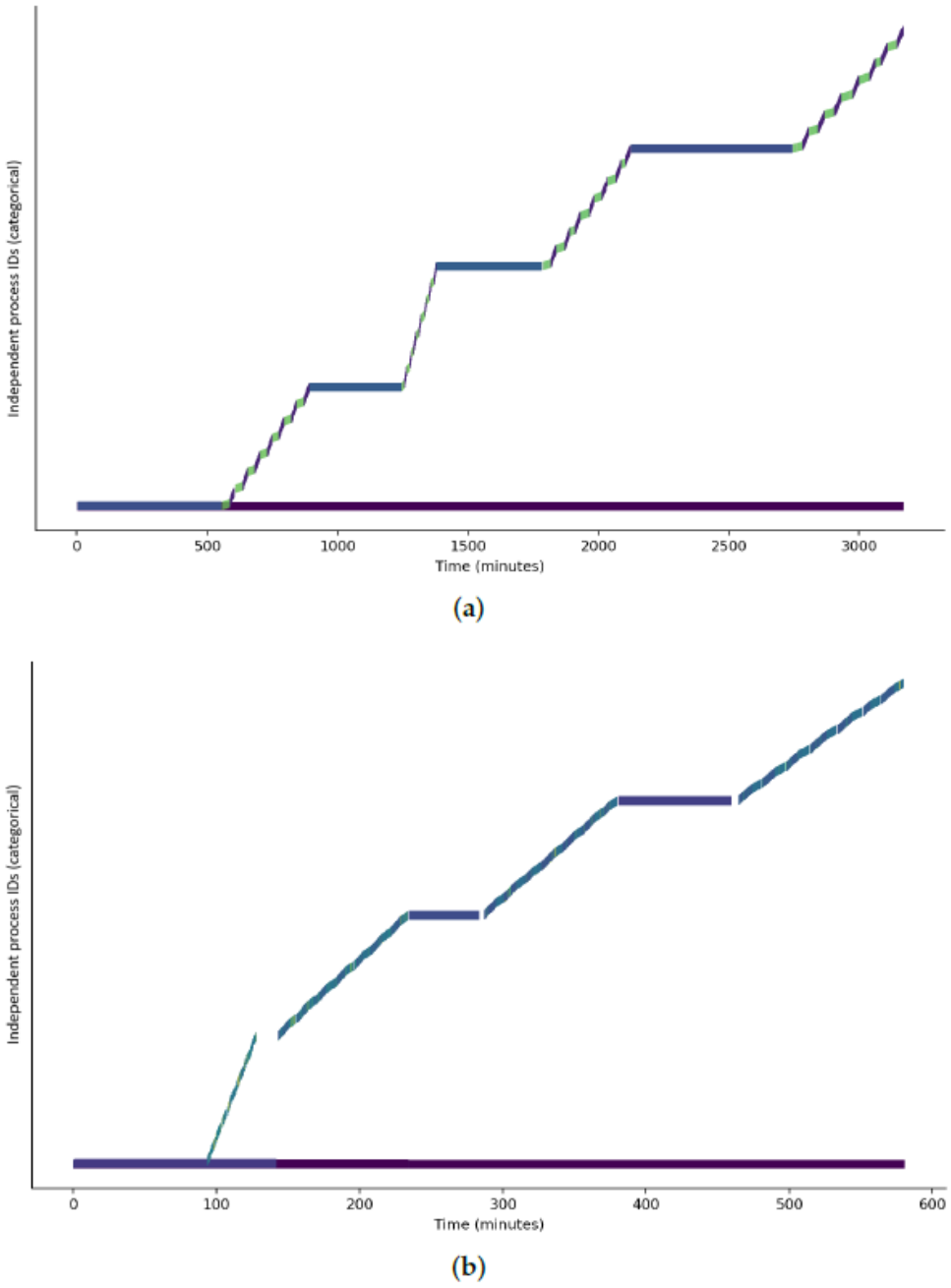


Figure 5.1 Execution concurrency profiles for sequential and parallel modes.

Worker lifetimes for the sequential (a) and parallel (b) configurations under identical runtime conditions. The y-axis is categorical and represents independent worker processes, while the x-axis shows wall-clock time in minutes from the start of the pipeline. Each colored segment indicates the active period of a worker, with color encoding its average CPU load (dark purple/blue = low, green = medium, yellow = high, using the same normalization in both panels). The long bar at the bottom represents the main controller process, which remains active throughout execution.

Resource utilization and execution dynamics are shown in Figure for the comparative analysis between the sequential emulation (**Figure 5.2a** – **Figure 5.2c**) and the adaptive parallel execution (**Figure 5.2b** – **Figure 5.2d**), highlighting clear differences in these aspects. **Figure 5.2a** and **Figure 5.2b** illustrates worker concurrency. In the parallel configuration, more workers remain active simultaneously, with pronounced overlapping lifetimes that reflect an efficient distribution of tasks across multiple processes. In contrast, the sequential emulation maintains a nearly constant and limited number of workers producing long plateaus of near-serial execution with minimal concurrency and limited exploitation of available CPU resources.

In parallel mode, CPU usage exhibited sustained peaks reaching several hundred percentage points, evidencing effective multi-core utilization, while memory usage remained stable within expected thresholds, indicating that overhead was well-managed despite higher concurrency (**Figure 5.2c** – **Figure 5.2d**). In sequential mode, CPU consumption stayed near the single-core baseline, with long inactive periods, and although memory usage was stable, the lack of concurrency restricted throughput. Each of these panels displays two curves for each metric: a solid blue line showing the raw one-second measurements and an orange dashed line representing a 15-second moving average that filters short-term fluctuations. This dual representation provides both instantaneous variations and smoothed trends, helping distinguish structural load patterns from transient spikes.

CPU and memory exhibit similar temporal shapes because both metrics respond directly to worker activity. As workers start, finish, or overlap, they jointly drive increases or decreases in CPU load and memory usage, resulting in curves with comparable forms. The difference lies in magnitude: CPU utilization is mapped to the left y-axis, and memory is mapped to the right y-axis. The colors do not distinguish between CPU and memory; instead, blue and orange encode raw and smoothed values, respectively, while the axes uniquely identify which metric is being displayed.

In parallel mode, workers become active immediately due to the initialization of process pools and task allocation structures, as reflected in **Figure 5.2b**. However, this startup phase produces only minimal CPU and memory activity, and therefore the raw and smoothed traces in **Figure 5.2d** remain close to zero during the first few minutes. This does not represent

missing data or misalignment; it simply reflects the low-cost initialization phase before sustained parallel computation begins

Taken together, these figures provide a comprehensive overview of the framework's performance. Concurrency plots **Figure 5.2a** and **Figure 5.2b** reveal differences in orchestration efficiency, while CPU and memory plots **Figure 5.2c** and **Figure 5.2d** quantify computational load and resource balance. The integration of these perspectives confirms that the parallel configuration not only reduces runtime but also maintains stable system performance, validating the design of the Multi-Cube framework for scalable, large-scale time series processing.

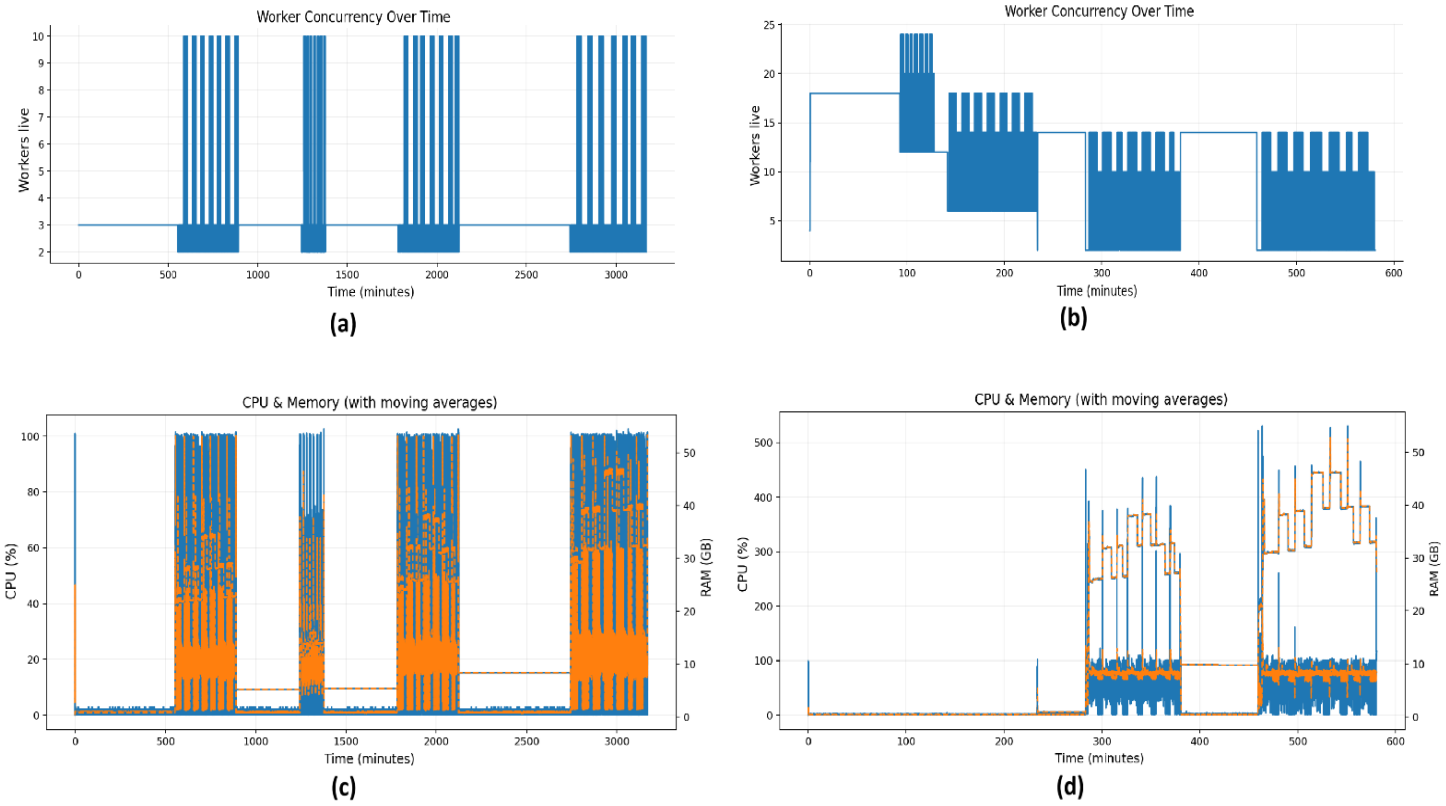
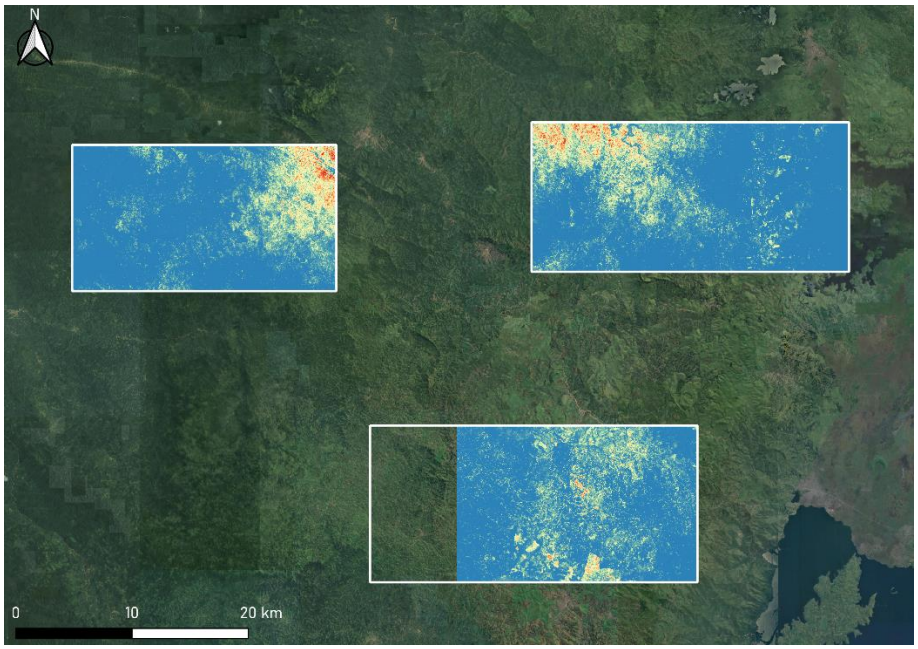


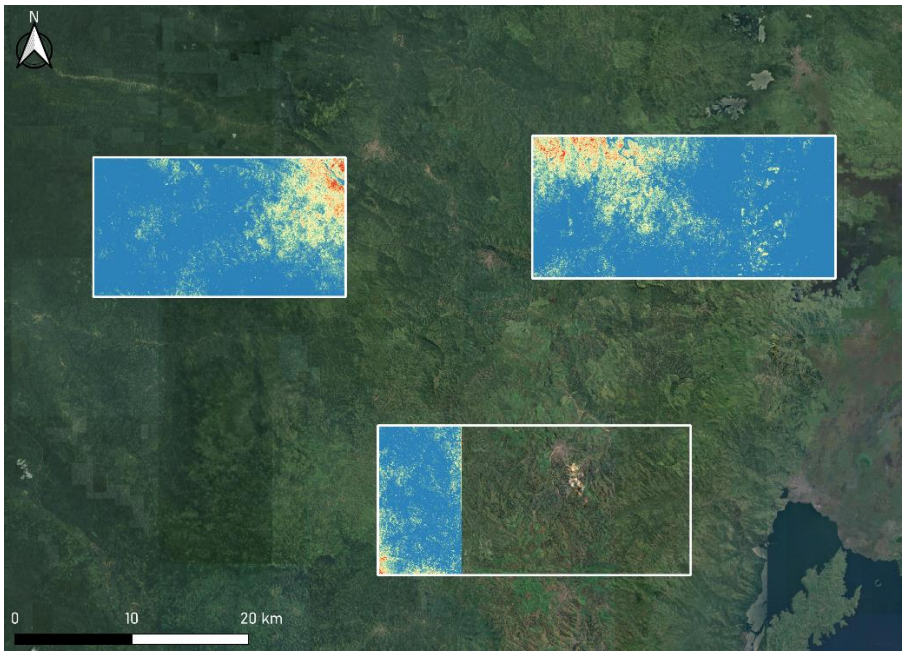
Figure 5.2 Sequential vs. adaptive parallel performance.
(a, b) Worker concurrency for sequential and parallel modes. (c, d) CPU and memory usage for sequential and parallel configurations.

Finally, in both the sequential emulation and the adaptive parallel execution modes, the framework successfully processed the three AOIs introduced in **Figure 4.1** validating its robustness under heterogeneous tile coverage scenarios. Figure presents the final outputs produced by the external algorithm within the Multi-Cube framework. **Figure 5.3a** and **Figure 5.3b** displays the three processed areas. The area with incomplete spatial coverage corresponds to Area C introduced in **Figure 4.1**. This apparent incompleteness is not due to missing data; rather, it reflects the automatic cube subdivision mechanism activated when none of the Sentinel-2 tiles provide full coverage of the original AOI, resulting in independent sub-cubes that together represent Area C. In this case, the Multi-Cube architecture automatically subdivided the AOI into two independent sub-series (C-1 and C-2), which were processed consistently in both sequential and parallel configurations, yielding independent results without user intervention.

In contrast, for Areas A and B introduced in **Figure 4.1** which were intersected by more than one tile on the same acquisition date, the framework systematically selected a single optimal tile, thereby preserving temporal integrity and avoiding redundant mosaicking. The results shown correspond to the application of the change detection algorithm by Deijns et al. The results are consistent with the expected behavior of Deijns et al.'s method. **Figure 5.3** illustrates how the automatic selection and subdivision logic of the Multi-Cube framework consistently produces coherent and reproducible time series suitable for large-scale analyses with algorithms operating within this framework.



(a)



(b)

Figure 5.3 Framework-driven spatial subdivision of an AOI

(a) Automatic subdivision of Area C into a spatially coherent sub-area (C-1), aligned with the native Sentinel-2 grid, as resolved by the framework due to incomplete single-tile coverage.

(b) Complementary sub-area (C-2) processed in parallel, ensuring full AOI coverage. Both panels are shown superimposed over a Google Earth basemap.

5.2 Scenario 2: Sub-Cube Access vs. Full-Cube Recalculation

In this experiment, it's compared on-demand extraction of a spatiotemporal sub-cube with a full reconstruction of a data cube for a newly requested area. The requested region corresponds to a small spatial subset (**Figure *5.4***) that is fully contained within Study Area A introduced in **Figure 4.1**. In addition, the request targets a restricted temporal window of one calendar year, covering the 12 months of 2023. As this temporal range is already included in the existing stable cube, the corresponding time series is readily available.

The marked performance advantage observed for the on-demand sub-cube extraction is a direct consequence of the framework's baseline-aware design. When both the requested spatial extent and temporal interval are already present in a stable cube, the framework bypasses the ingestion and preprocessing stages and accesses only the minimal set of required spatiotemporal chunks. This approach avoids unnecessary cube reconstruction and substantially reduces computational overhead.

As a result, on-demand sub-cube extraction significantly lowers both I/O operations and CPU usage relative to a full cube rebuild, explaining the pronounced performance gap observed in this experiment.

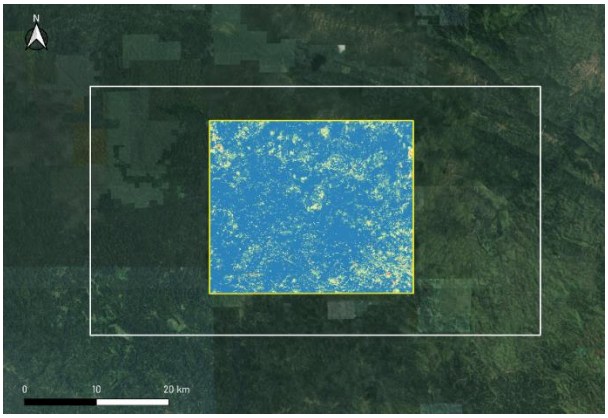


Figure 5.4 On-demand sub-cube extraction within a study area

The white polygon corresponds to Study Area A, whose full spatial footprint is shown to contextualize the location of the extracted subset. The yellow inner extent represents the sub-cube extracted by the framework. It is fully contained within the spatial limits of Study Area A and was used for the temporal analysis window. Google Earth basemap.

For this case, has been evaluated three configurations:

- Full-Cube Recalculation (Sequential Emulation), which rebuilds the entire pipeline for the AOI from ingestion to analysis, assuming no prior data exist
- Full-Cube Recalculation (Optimized Parallel Execution), which rebuilds the entire AOI from scratch under the framework's optimized parallel configuration
- On-Demand Sub-Cube Access, which avoids cube reconstruction by materializing an in-memory spatiotemporal sub-cube directly from the stable cube containing the requested area, writing only the analysis artifacts and leaving the dynamic baseline unchanged.

The framework automatically resolves the source data through spatiotemporal intersection, requiring only a target sub-area and temporal range from the user. When the requested region is already contained within an existing stable cube, the system restricts I/O to the corresponding spatial and temporal chunks instead of accessing the full dataset. This selective access avoids the expensive stages associated with monolithic rebuilds, reducing latency, read/write volume and CPU usage. The objective of this experiment was to quantify the resulting time until a result was acquired and resource savings relative to full cube reconstruction.

Table 6 compares the three configurations: Sequential Rebuild, Parallel Rebuild and On-Demand Sub-Cube and highlights the clear performance gains obtained when the requested sub-area is fully contained, both spatially and temporally, within an existing stable cube. The On-Demand configuration achieved the shortest wall-clock time (approximately 16.2 minutes), outperforming the Parallel Rebuild (approximately 22.9 minutes) and the Sequential Rebuild (approximately 135 minutes). This result corresponds to a speed-up of about 8.3× relative to the Sequential Rebuild and 1.4× relative to the Parallel Rebuild. These runtime improvements were accompanied by a drastic reduction in write I/O, with a value of only 9.3 MB for the On-Demand Sub-Cube, while those for the Sequential and Parallel Rebuilds were 2.3 GB and 2.1 GB, respectively, representing reductions of roughly 248× and 221×, reflecting the fact that the On-Demand strategy reuses the existing stable cube and rather than rebuilding the baseline, retains only the analysis artifacts.

CPU utilization also reflects this efficiency: the On-Demand Sub-Cube maintained the highest mean CPU usage (approximately 69.6 %), which can be compared with the values for Parallel Rebuild (approximately 45.8 %) and Sequential Rebuild (approximately 7.7 %), while peak RAM usage remained comparable across runs (approximately 3.3–4.1 GB). Overall, when the analysis targets a sub-area already contained spatiotemporally within a stable cube, on-demand sub-cube extraction avoids repeated reconstruction, drastically reduces write I/O, and shortens the time until a result is obtained without increasing memory requirements.

Metric	Sequential Rebuild	Parallel Rebuild	On Demand Sub-Cube
Elapsed (H:M:S)	02:14:57	00:22:56	00:16:14
Mean CPU (%)	7.65	45.79	69.61
RAM peak (GB) (%)	3.35	4.09	3.69
Disk read (MB, cum.)	216.1	21.6	1501.2
Disk write (MB, cum.)	2304.0	2057.0	9.3

Table 6 Comparative summary of performance (Sub-cube)

Across the three configurations, the Disk I/O profiles in **Figure 5.5** reveal a clear discontinuity between the rebuild-based workflows and the On-Demand Sub-Cube strategy. Both the Sequential **Figure 5.5b** and Parallel **Figure 5.5a** rebuild configurations generate 2.1- 2.3~GB of cumulative disk writing, meaning that the entire cube is rewritten regardless of the requested extent. The Parallel mode improves runtime but not efficiency, as the I/O footprint remains unchanged. Conversely, the On-Demand Sub-Cube (**Figure 5.5c**) collapses the write curve to almost zero (9~MB, approx. 248× reduction) because it reuses existing baseline cubes and writes only localized outputs instead of rebuilding the full archive. This transforms the process from a heavy storage-bound workflow into a lightweight and near-interactive extraction operation suitable for iterative or localized analyses.

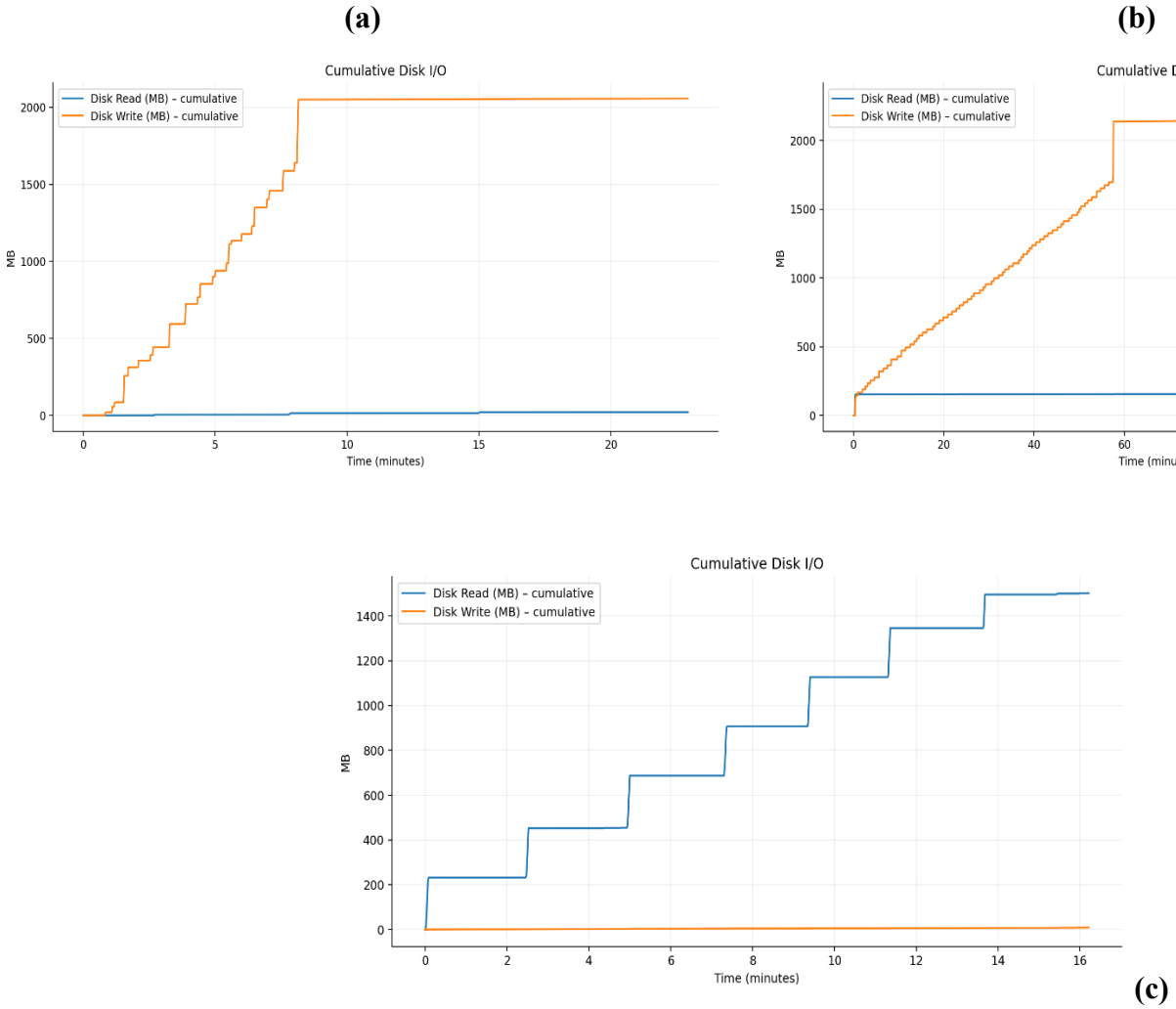


Figure 5.5 I/O footprints of rebuild-centric and on-demand execution.
(a) Parallel Rebuild: Full-cube recomputation with a high I/O footprint. **(b)** Sequential Rebuild: Full-cube recomputation with similar disk-writing accumulation over a longer runtime. **(c)** On Demand Sub-Cube: Minimal writing activity restricted to the requested spatial and temporal subset

Finally, **Figure 5.6** is the worker-level CPU heatmap for the On-Demand Sub-Cube configuration; it shows a distinctive execution pattern characterized by the selective activation of worker processes (PIDs) rather than continuous load distribution. Each PID represents a separate worker process, and only a single high-intensity burst of CPU activity can be observed, likely corresponding to the precise moment when the sub-cube extraction or analysis detection stage was triggered. Aside from this isolated event, the remaining time bins exhibit near-idle CPU levels, indicating that no persistent background workers were

maintained. This behavior confirms that unlike full-rebuild workflows that continuously engage in multiple processes throughout the execution window, the On-Demand strategy allocates computational resources only at the exact moment of need, avoiding redundant cube-wide reconstruction and resulting in a highly efficient demand-driven processing profile. Together with the I/O and runtime results, this finding confirms that on-demand sub-cube access fundamentally shifts the execution model from archive-wide processing to targeted, responsive analysis.

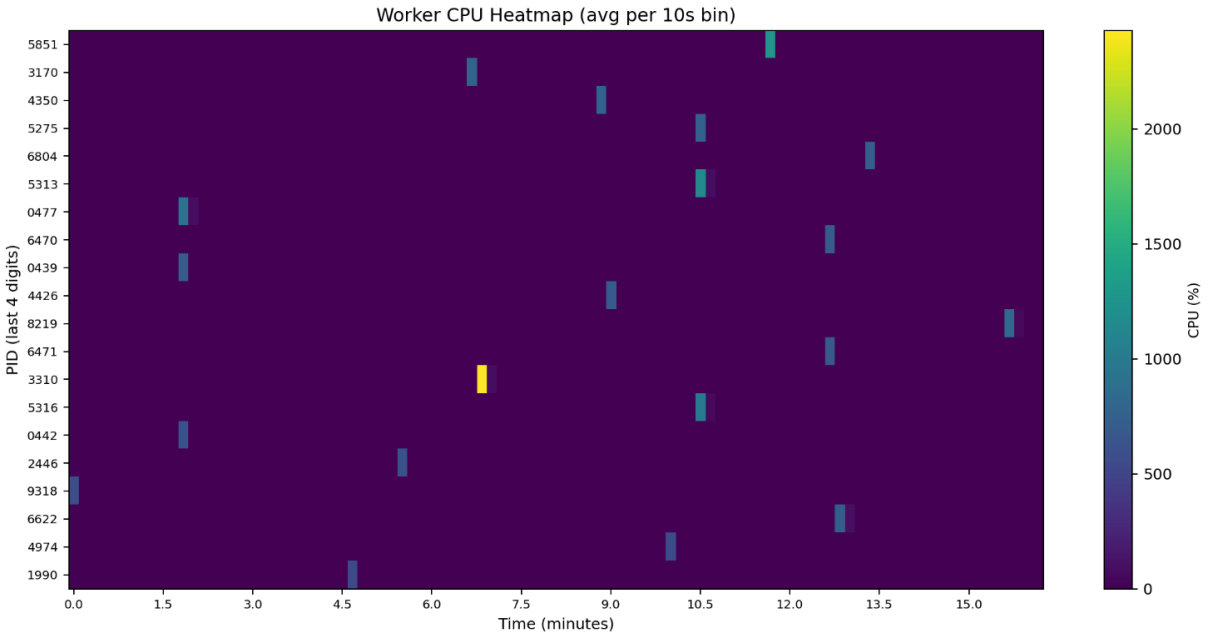


Figure 5.6 CPU heatmap for the On-Demand Sub-Cube configuration

A brief, localized activation of worker PIDs occurs during the sub-cube request, showing the demand-driven nature of this mode, which is unlike the continuous load observed in full-rebuild workflows.

In summary, the performance evaluation demonstrates that the proposed Multi-Cube framework not only scales efficiently under full reconstruction scenarios but can also transition to a lightweight, demand-driven execution mode when prior baselines are available. By combining incremental storage logic with selective sub-cube access, the architecture moves away from a monolithic rebuild paradigm toward an adaptive execution model that optimizes both turnaround time and resource utilization.

5.3 Architectural Discussion of the Results

The experimental results presented in the previous sections provide empirical evidence of how the proposed Multi-Cube framework behaves under different execution and update strategies, and allow a direct comparison with traditional rebuild-centric processing approaches. This discussion focuses on interpreting these results from an architectural perspective, emphasizing how the observed performance gains emerge as a consequence of persistent data structures, policy-driven orchestration, and incremental baseline management, rather than from isolated algorithmic optimizations.

5.3.1 Execution behavior and adaptive parallelization

The comparison between sequential emulation and adaptive parallel execution (Scenario 1) highlights that parallelism in the Multi-Cube framework is not an externally imposed configuration but an intrinsic property of the orchestration logic. As shown by the worker lifetime and CPU utilization profiles, the sequential mode effectively establishes a lower-bound execution baseline, characterized by short-lived, mostly non-overlapping worker processes and limited temporal concurrency. This behavior closely resembles classical file-centric pipelines, where tasks are executed in near-serial order even when computational resources are available (Simões et al., 2021).

In contrast, the adaptive parallel configuration exhibits sustained concurrency, continuous worker replacement, and stable plateaus of simultaneous task execution. Importantly, these execution patterns emerge without requiring user intervention or workflow restructuring. The framework dynamically allocates workers according to the detected workload and available resources, confirming that parallelism is treated as an execution strategy rather than a design constraint. This behavior directly addresses limitations identified in execution-centric architectures, where parallelism is typically hard-coded into workflows or requires explicit user configuration (Chapter 2; Montero et al., 2024)

From an architectural standpoint, these results demonstrate that the proposed framework decouples analytical intent from execution mechanics. Users specify *what* data are required and *which* update policy applies, while the system autonomously determines *how* tasks are

executed. This separation is critical for recurrent monitoring scenarios, where execution conditions may vary over time and across deployment environments.

5.3.2 Rebuild-centric processing versus incremental baseline reuse

Scenario 2 provides the strongest evidence supporting the central thesis of this work: that persistent, state-aware data representations fundamentally alter the cost structure of satellite time-series processing. Both the parallel and sequential rebuild configurations exhibit comparable cumulative disk-writing volumes, despite differences in wall-clock runtime. This result confirms that traditional rebuild-centric pipelines remain I/O-bound, as the entire cube is regenerated regardless of the spatial or temporal scope of the request. Parallel execution reduces runtime but does not mitigate redundant data movement, reinforcing observations reported in previous studies on large-scale EO workflows (Montero et al., 2024; Karasante et al. 2025)

By contrast, the incremental reuse strategy drastically reduces disk I/O by appending only previously unseen observations or by reusing existing baseline data. The resulting reduction in write volume, by more than two orders of magnitude, illustrates that performance gains stem primarily from avoiding unnecessary recomputation rather than from faster execution alone. This behavior directly validates the design principles introduced in Chapter 3, where stable data cubes are treated as persistent computational artifacts rather than transient workflow outputs.

These findings confirm that incremental baseline reuse shifts the dominant cost from repeated data materialization to selective access and lightweight updates. As a result, the marginal cost of extending an existing time series becomes proportional to the volume of genuinely new data, rather than to the total archive size, aligning with the architectural objectives formulated in Chapter 1.

5.3.3 On-demand sub-cube access as a consequence of persistence

Within the same scenario, the on-demand sub-cube access mode illustrates a fundamentally different execution path enabled by the incremental architecture. Unlike rebuild-centric workflows, which must regenerate full cubes even for localized queries, the framework

resolves existing stable cubes and extracts only the requested spatial and temporal subsets in memory. This operation is strictly read-only with respect to the persistent baseline and therefore incurs negligible disk-writing overhead.

Crucially, this behavior should not be interpreted as a standalone optimization technique. Instead, it is a direct consequence of maintaining persistent, chunked data structures that preserve spatial and temporal coherence over time. The ability to assemble sub-cubes on demand without modifying the stored baseline demonstrates that the framework treats historical data as a reusable asset rather than as an intermediate product. Similar limitations in traditional EO infrastructures have been identified in the literature, where partial analyses still require full archive reconstruction due to the lack of persistent state (Munteanu, 2024).

From an architectural perspective, the on-demand mode transforms localized analyses from heavy, storage-bound workflows into lightweight extraction operations, making near-interactive exploration feasible even for large and long-term EO archives. This capability is particularly relevant for exploratory and iterative monitoring scenarios, where analysts repeatedly query overlapping spatial domains as new observations become available.

5.3.4 Architectural implications and synthesis

Taken together, the results demonstrate a clear transition from rebuild-first, execution-centric processing toward a reuse-first, state-aware architectural model. The observed reductions in runtime and disk I/O are not isolated performance improvements but emergent properties of an architecture that explicitly reasons about data persistence, temporal completeness, and execution context.

Unlike traditional pipelines, where each request triggers a full recomputation regardless of prior work, the Multi-Cube framework embeds decision-making mechanisms that determine whether data should be appended, reused or accessed on demand, a limitation widely reported in execution-centric EO processing architectures (Montero et al., 2024). This behavior directly addresses the structural limitations discussed in Chapter 2 and provides empirical validation for the architectural choices introduced in Chapter 3.

Overall, the experimental results confirm that intelligent orchestration combined with incremental data management enables scalable and sustainable processing of large EO time series. Rather than optimizing individual workflow components, the proposed approach redefines how computation is organized around evolving datasets, an architectural shift increasingly recognized as necessary for long-term EO data exploitation (Munteanu, 2024), offering a robust foundation for recurrent and long-term EO monitoring.

5.4 Advantages, Limitations, and Trade-offs of the Approach

The proposed Multi-Cube framework introduces a set of architectural mechanisms specifically designed to support recurrent and large-scale EO time-series processing. While the experimental results demonstrate substantial gains in execution efficiency and data reuse, it is essential to critically assess both the advantages and the inherent limitations of the approach, as well as the trade-offs implied by its design choices. This section provides a balanced discussion aimed at contextualizing the framework's applicability and scope.

5.4.1 Advantages of the Proposed Approach

One of the primary advantages of the Multi-Cube architecture lies in its explicit treatment of persistence and state as first-class architectural concepts. Unlike traditional execution-centric EO pipelines, where each request is handled as an independent workflow invocation, the proposed framework maintains stable, reusable data cubes that evolve incrementally over time, a requirement increasingly emphasized in large-scale, long-term EO data cube initiatives (Chen et al., 2024). This design directly enables efficient recurrent monitoring, where repeated analyses over the same spatial domains are common, such as in global or continental-scale monitoring datasets built from multi-decadal EO archives (Cai et al., 2025).

The experimental results show that this persistence-driven strategy significantly reduces redundant computation and disk I/O, particularly when new observations are incrementally incorporated into existing time series. By appending only unseen dates or selectively reusing historical data, the framework ensures that the marginal cost of extending an archive scales with newly ingested data rather than with the full temporal depth of the dataset. This behavior is particularly advantageous for long-term monitoring applications, where archives may span

multiple years or decades.

Another key advantage is the decoupling between analytical intent and execution strategy. Users interact with the system by defining AOIs, temporal ranges and update policies, while execution details, such as parallelization, task scheduling, and resource allocation, are resolved internally by the orchestrator. This abstraction reduces the need for domain users to manage low-level computational concerns and mitigates the risk of inefficient workflow configurations, a limitation frequently observed in execution-centric architectures (Montero et al., 2024; Karasante et al. 2025)

Finally, the support for on-demand sub-cube access represents a practical advantage for exploratory and iterative analyses. By enabling read-only extraction of localized spatiotemporal subsets without modifying the persistent baseline, the framework allows near-interactive access patterns that are difficult to achieve with rebuild-centric pipelines. This capability broadens the range of analytical workflows that can be supported efficiently, from large-scale batch processing to localized temporal investigations.

5.4.2 Architectural Limitations

Despite these advantages, the proposed framework also exhibits several limitations that should be explicitly acknowledged. A fundamental assumption of the Multi-Cube approach is the existence of stable spatial units that can be deterministically associated with persistent data cubes. In the current implementation, this association is achieved through geometry-based spatial hashing. While effective for recurrent monitoring over predefined AOIs, this design may be less suitable for highly dynamic spatial queries or continuously shifting regions of interest, where persistent spatial identities are harder to maintain.

Another limitation concerns the initial cost of cube construction. Although incremental updates substantially reduce the cost of extending an existing archive, the first-time creation of a stable cube still requires full preprocessing of all contributing observations. For very large AOIs or long historical archives, this initial investment can be significant, even if it is amortized over subsequent reuse. This behavior reflects an inherent trade-off between upfront preprocessing and long-term efficiency.

The framework deliberately introduces additional persistent storage compared to purely transient processing pipelines. This storage is used to maintain stable Zarr-based cubes and per-date intermediate representations that enable incremental updates, data reuse, and on-demand sub-cube access. While such persistence requires additional disk space, it is a conscious architectural trade-off that significantly reduces redundant recomputation and disk I/O in recurrent monitoring scenarios. This design choice may be less suitable for storage-constrained or fully ephemeral deployments, but is well aligned with long-term and operational EO monitoring contexts.

From a multisensor perspective, the proposed framework is sensor-agnostic at the architectural level, as its mechanisms for spatial subdivision, cube construction, baseline management, and execution orchestration do not depend on any specific sensor. The current implementation assumes that input datasets are provided in a geometrically and radiometrically consistent form suitable for spatiotemporal integration. When this condition is not met, additional preprocessing or calibration steps may be required prior to ingestion. Such steps are external to the core architecture and can be incorporated as modular preprocessing components without affecting the framework's design. In practice, this assumption is naturally satisfied by sensors delivering analysis-ready products, but it does not restrict the framework to a particular mission.

5.4.3 Trade-offs and Design Decisions

The limitations outlined above are not accidental shortcomings but reflect deliberate architectural trade-offs. By prioritizing persistent data representations and reuse-first execution, the framework accepts higher storage requirements and initial preprocessing costs in exchange for substantially improved efficiency in recurrent and long-term analyses. This trade-off is particularly favorable in operational monitoring scenarios, where the same AOIs are queried repeatedly as new data arrive.

Similarly, the reliance on stable spatial units simplifies cube management and indexing while constraining flexibility in handling arbitrary spatial queries. This design choice reflects a conscious emphasis on monitoring-oriented workflows rather than ad hoc exploratory analyses over constantly changing spatial extents.

Another key trade-off concerns execution behavior. While adaptive parallelization improves throughput and resource utilization, it introduces variability in execution patterns depending on available hardware and workload characteristics. Although this variability is handled transparently by the orchestrator, it may complicate reproducibility at the level of exact execution traces, even when analytical results remain consistent.

Overall, the Multi-Cube framework embodies a shift from simplicity of execution toward architectural intelligence and persistence. Rather than minimizing system state, it leverages controlled statefulness to reduce redundancy, improve scalability and support evolving EO time series. These trade-offs define the applicability envelope of the approach and should be carefully considered when selecting or extending the framework for specific operational contexts.

CHAPTER 6

Conclusions and future work

6 Conclusions and Future Work

6.1 General Conclusions

This thesis addressed a central limitation of current EO processing infrastructures: the lack of architectural mechanisms capable of sustaining recurrent and long-term satellite time-series analysis without incurring systematic recomputation and excessive data movement. While significant advances have been made in scalable storage, cloud-native formats, and distributed execution, most existing approaches remain fundamentally execution-centric, treating each processing request as an isolated event rather than as part of an evolving analytical context.

To address this gap, this work proposed and validated a Multi-Cube incremental architecture designed to manage large EO time series through persistent, state-aware data structures and policy-driven orchestration. Rather than focusing on algorithmic acceleration, the framework redefines how computation is organized around satellite archives by explicitly modeling spatial persistence, temporal continuity, and execution decisions as first-class architectural concerns.

The experimental evaluation demonstrated that this architectural shift leads to substantial reductions in both computational cost and disk I/O. Sequential and parallel execution experiments showed that adaptive parallelization emerges naturally from the orchestration logic without requiring manual intervention or workflow redesign. More importantly, comparisons between rebuild-centric processing and incremental reuse confirmed that the dominant performance gains originate from avoiding redundant recomputation and from reusing stable baselines rather than from faster execution alone.

The results further showed that on-demand sub-cube access is not an isolated optimization but a direct consequence of maintaining persistent, chunked data representations. This capability enables localized and exploratory analyses to be performed with negligible write overhead, transforming traditionally heavy, storage-bound workflows into lightweight extraction operations. Collectively, these findings validate the central hypothesis of this thesis: that persistent, incremental, and decision-driven architectures provide a more scalable

and sustainable foundation for long-term EO monitoring than traditional rebuild-first processing models.

6.2 Contributions of the Thesis

This thesis makes several contributions at the architectural and methodological levels to the field of large-scale EO data processing.

First, it introduces a **Multi-Cube architectural model** that departs from monolithic data cube designs by generating and managing multiple stable, spatially bounded cubes that evolve independently over time. This approach enables scalable subdivision, efficient reuse, and deterministic access to recurrent areas of interest without relying on centralized metadata catalogs or global rebuilds.

Second, the thesis formalizes the concept of a **dynamic baseline strategy** governed by explicit update policies. By distinguishing between auto-append, append-from-missing, recompute-window, and force-rebuild behaviors, the framework provides a principled mechanism for deciding when data should be incrementally extended, selectively accessed, or fully regenerated. This policy-driven logic transforms baseline management from an implicit workflow assumption into an explicit architectural component.

Third, the work proposes an **intelligent orchestration mechanism** that decouples analytical intent from execution strategy. Users specify spatial and temporal requirements, while the framework autonomously resolves spatial subdivision, task scheduling and adaptive parallelization. This separation reduces user burden and mitigates common sources of inefficiency associated with manual workflow configuration.

Fourth, the thesis demonstrates how **persistent, chunked data representations** enable on-demand sub-cube extraction without modifying historical baselines. This contribution highlights a practical pathway for supporting near-interactive analysis over large EO archives while preserving long-term consistency.

Finally, the thesis provides **empirical evidence** showing that architectural design choices—particularly persistence and reuse-first execution—can yield orders of magnitude reductions

in disk I/O and significant runtime improvements in recurrent monitoring scenarios. These results reinforce the importance of architectural reasoning in EO systems, complementing ongoing advances in algorithms, machine learning and cloud infrastructure.

6.3 Future Work Directions

While the proposed framework establishes a robust foundation for incremental EO time-series processing, several avenues for future research and development naturally emerge from this work.

A relevant direction for future work concerns the extension of the framework toward more **advanced and automated multisensor standardization workflows**. While the proposed architecture is multisensor by design and remains agnostic to the specific origin of the input data, further developments could focus on automating sensor-specific normalization and harmonization steps when required. Such mechanisms would operate as modular preprocessing components, preserving the architectural decoupling between core cube management and sensor-dependent corrections, and would increase the framework's applicability to highly heterogeneous EO datasets.

A second avenue involves **adaptive chunking and storage optimization**. The current implementation relies on predefined chunking strategies optimized for general access patterns. Future research could explore dynamic or workload-aware chunking schemes that adapt to observed query behavior, further improving access efficiency and resource utilization.

Another relevant extension relates to **distributed and cloud-native deployments**. While the framework is compatible with cloud-native storage formats, future work could investigate deeper integration with object storage backends, serverless execution environments and federated data cube infrastructures spanning multiple geographic regions or administrative domains.

From an analytical perspective, integrating **advanced analytical and machine learning modules** on top of the proposed architecture constitutes a natural next step. The persistent and incremental nature of the Multi-Cube framework provides a suitable substrate for

temporal modeling, anomaly detection and learning-based approaches that benefit from stable historical baselines and efficient incremental updates.

Finally, future work could explore **formal performance modeling and cost prediction**, enabling the framework to anticipate computational and storage costs associated with different update policies and execution strategies. Such capabilities would further enhance decision-making at the orchestration level and support resource-aware planning in operational settings.

7 BIBLIOGRAPHY

Abernathey, R. P., Augspurger, T., Banihirwe, A., Blackmon-Luca, C. C., Crone, T. J., Gentemann, C. L., Hamman, J. J., Henderson, N., Lepore, C., McCaie, T. A., Robinson, N. H., & Signell, R. P. (2021). Cloud-native repositories for big scientific data. *Computing in Science & Engineering, 23*(2), 26–35.

Adegun, A. A., Viriri, S., & Tapamo, J. R. (2023). Review of deep learning methods for remote sensing satellite image classification: Experimental survey and comparative analysis. *Journal of Big Data, 10*, Article 93.

Agram, P. S., Warren, M. S., Calef, M. T., & Arko, S. A. (2022). An efficient global-scale Sentinel-1 radar backscatter and interferometric processing system. *Remote Sensing, 14*(15), Article 3524.

Appel, M., & Pebesma, E. (2019). On-demand processing of data cubes from satellite image collections with the gdalcubes library. *Data, 4*(3), Article 92.

Augustin, H., Sudmanns, M., Tiede, D., Lang, S., & Baraldi, A. (2019). Semantic Earth observation data cubes. *Data, 4*, Article 102.

Baumann, P., Misev, D., Merticariu, V., & Pham Huu, B. H. (2021). Array databases: Concepts, standards, implementations. *Big Data Journal, 8*, 28–44.

Baumann, P., Misev, D., Merticariu, V., & Pham Huu, B. (2018). Datacubes: Towards space/time analysis-ready data. In J. Döllner, M. Jobst, & P. Schmitz (Eds.), *Service-oriented mapping – Changing paradigms in map production and geoinformation management* (pp. 269–299). Springer.

Burgueño, A. M., Aldana-Martin, J. F., Vazquez-Pendon, M., Barba-Gonzalez, C., Jimenez Gomez, Y., Garcia Millan, V., & Navas-Delgado, I. (2023). Scalable approach for high-resolution land cover: A case study in the Mediterranean Basin. *Journal of Big Data, 10*, Article 91.

Cai, Y., Li, X., Zhu, P., Nie, S., Wang, C., Liu, X., & Chen, Y. (2025). China Earth observation data cube: The 30-m seamless annual leaf-on Landsat composites from 1985–2023. *Remote Sensing, 5*, 0698.

Claverie, M., Ju, J., Masek, J. G., Dungan, J. L., Vermote, E. F., Roger, J. C., Skakun, S. V., & Justice, C. (2018). The harmonized Landsat and Sentinel-2 surface reflectance data set. *Remote Sensing of Environment, 219*, 145–161.

Committee on Earth Observation Satellites (CEOS) Land Surface Imaging Virtual Constellation (LSI-VC). (2024). *CEOS analysis ready data interoperability handbook* (Version 2.0). Committee on Earth Observation Satellites.

Chen, S., Wang, J., Liu, Q., Liang, X., Liu, R., Qin, P., Yuan, J., Wei, J., Yuan, S., Huang, H., & Gong, P. (2024). Global 30 m seamless data cube (2000–2022) of land surface reflectance generated from Landsat 5, 7, 8, and 9 and MODIS Terra constellations. *Earth System Science Data, 16*, 5449–5475.

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., Ferreira da Silva, R., Livny, M., & Wenger, K. (2015). Pegasus, a workflow management system for science automation. *Future Generation Computer Systems, 46*, 17–35.

Deijns, A. A. J., Michéa, D., Déprez, A., Malet, J.-P., Kervyn, F., Thiery, W., & Dewitte, O. (2024). A semi-supervised multi-temporal landslide and flash flood event detection methodology using massive satellite image time series. *ISPRS Journal of Photogrammetry and Remote Sensing, 215*, 400–418.

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., & Meygret, A. (2012). Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sensing of Environment, 120*, 25–36.

Dritsas, E., & Trigka, M. (2025). Remote sensing and geospatial analysis in the big data era: A survey. *Remote Sensing, 17*, 550.

Durbha, S., Sanyal, J., Yang, L., Chaudhari, S., Bhangale, U., Bharambe, U., & Kurte, K. (2023). *Advances in scalable and intelligent geospatial analytics: Challenges and applications*. CRC Press.

Enache, S., Louis, J., Pflug, B., de los Reyes, R., Lafrance, B., Clerc, S., Barrot, G., Alhammoud, B., Poustomis, F., Iannone, R. Q., Bruniquel, J., Boccia, V., & Gascon, F. (2023). Copernicus Sentinel-2 Collection-1: A consistent dataset of multispectral imagery with enhanced quality. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 4503–4506). Pasadena, CA, United States.

Fang, F., Zhou, T., Song, Z., & Lu, J. (2023). MMCAN: Multi-modal cross-attention network for free-space detection with uncalibrated hyperspectral sensors. *Remote Sensing, 15*, Article 1142.

Ferreira, K. R., Queiroz, G. R., Marujo, R. F. B., & Costa, R. W. (2022). Building Earth observation data cubes on AWS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B3-2022*, 597–604.

Fu, Y., Zhu, Z., Liu, L., Zhan, W., He, T., Shen, H., Zhao, J., Liu, Y., Zhang, H., Liu, Z., Xue, Y., & Ao, Z. (2024). Remote sensing time series analysis: A review of data and applications. *Journal of Remote Sensing, 4*, Article 0285.

Gaigalas, J., Di, L., & Sun, Z. (2019). Advanced cyberinfrastructure to enable search of big climate datasets in THREDDS. *ISPRS International Journal of Geo-Information, 8*(11), Article 494.

Gaveau, D. L. A., Descals, A., Salim, M. A., Sheil, D., & Sloan, S. (2021). Refined burned-area mapping protocol using Sentinel-2 data increases estimate of 2019 Indonesian burning. *Earth System Science Data, 13*, 5353–5368.

Giuliani, G., Camara, G., Killough, B., & Minchin, S. (2019). Earth observation open science: Enhancing reproducible science using data cubes. *Data, 4*(4), Article 147.

Giuliani, G., Chatenoux, B., De Bono, A., Rodila, D., Richard, J.-P., Allenbach, K., … Peduzzi, P. (2017). Building an Earth observations data cube: Lessons learned from the Swiss

Data Cube. *Big Earth Data, 1*(1–2), 100–117.

Giuliani, G.; Maso, J.; Mazzetti, P.; Nativi, S.; Zabala, A. Paving the Way to Increased Interoperability of Earth Observations Data Cubes. Data. 2019, 4, 113

Gomes, V. C. F., Queiroz, G. R., & Ferreira, K. R. (2020). An overview of platforms for big Earth observation data management and analysis. *Remote Sensing, 12*, Article 1253.

Gómez-Chova, L., Tuia, D., Moser, G., & Camps-Valls, G. (2015). Multimodal classification of remote sensing images: A review and future directions. *Proceedings of the IEEE, 105*(8), 1560–1584.

Guo, J., Huang, C., & Hou, J. (2022). A scalable computing resources system for remote sensing big data processing using GeoPySpark based on Spark on Kubernetes. *Remote Sensing, 14*(3), Article 521.

Iacono, L. L., Gaitán, A. L., Giraldo, F. X., & Ibarrola, L. (2023). SNDVI: A new scalable serverless framework to compute NDVI. *Frontiers in High Performance Computing, 2*, Article 1151530.

Jensen, J. R. (2016). *Introductory digital image processing: A remote sensing perspective* (4th ed.). Pearson Education.

Ju, J., Zhou, Q., Freitag, B., Roy, D. P., Zhang, H., Sridhar, M., Mandel, J., Arab, S., Schmidt, G. L., Crawford, C. J., Gascon, F., Strobl, P. A., Masek, J. G., & Neigh, C. S. R. (2025). The harmonized Landsat and Sentinel-2 version 2.0 surface reflectance dataset. *Remote Sensing of Environment, 324*, Article 114723.

Karasante, I., Alonso, L., Prapas, I., Ahuja, A., Carvalhais, N., & Papoutsis, I. (2025). SeasFire Cube: A multivariate dataset for global wildfire modeling. *Scientific Data, 12*, Article 368.

Kempeneers, P., Kliment, T., Marletta, L., & Soille, P. (2022). Parallel processing strategies for geospatial data in a cloud computing infrastructure. *Remote Sensing, 14*(2), Article 398.

Killough, B. (2018). Overview of the Open Data Cube initiative. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2018)* (pp. 8629–8632).

Kröber, F., Sudmanns, M., Abad, L., & Tiede, D. (2025). On-demand, semantic EO data cubes – Knowledge-based, semantic querying of multimodal data for mesoscale analyses anywhere on Earth. *ISPRS Journal of Photogrammetry and Remote Sensing, 228*, 552–565.

Lawler, S., Williams, T., Lehman, W., Lindemer, C., Rosa, D., Ferreira, C., & Zhang, C. (2025). Evaluation of the spatiotemporal asset catalog for management and discovery of FAIR flood hazard models. *Environmental Modelling & Software, 183*, Article 106230.

Lewis, A., Oliver, S., Lymburner, L., Evans, B., Wyborn, L., Mueller, N., Raevksi, G., Hooke, J., Woodcock, R., Sixsmith, J., Wu, W., Tan, P., Li, F., Killough, B., Minchin, S., Roberts, D., Ayers, D., Bala, B., Dwyer, J., Dekker, A., Dhu, T., Hicks, A., Ip, A., Purss, M., Richards, C., Sagar, S., Trenham, C., Wang, P., & Wang, L.-W. (2017). The Australian geoscience data cube—Foundations and lessons learned. *Remote Sensing of Environment, 202*, 276–292.

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2015). *Remote sensing and image interpretation* (7th ed.). Wiley.

Mahecha, M. D., Gans, F., Brandt, G., Christiansen, R., Cornell, S. E., Fomferra, N., Kraemer, G., Peters, J., Bodesheim, P., Camps-Valls, G., Donges, J. F., Dorigo, W., Estupinan-Suarez, L. M., Gutierrez-Velez, V. H., Gutwin, M., Jung, M., Londoño, M. C., Miralles, D. G., Papastefanou, P., & Reichstein, M. (2020). Earth system data cubes unravel global multivariate dynamics. *Earth System Dynamics, 11*(1), 201–234.

Michels, A., Padmanabhan, A., Xiao, Z., Kotak, M., Baig, F., & Wang, S. (2024). CyberGIS-Compute: Middleware for democratizing scalable geocomputation. *SoftwareX, 26*, Article 101691.

Montero, D., Kraemer, G., Anghelea, A., Aybar, C., Brandt, G., Camps-Valls, G., Cremer, F., Flik, I., Gans, F., Habershon, S., Ji, C., Kattenborn, T., Martínez-Ferrer, L., Martinuzzi, F., Reinhardt, M., Söchting, M., Teber, K., & Mahecha, M. D. (2024). Earth system data cubes: Avenues for advancing Earth system research. *Environmental Data Science, 3*, Article e27.

Munteanu, A. (2024). Data cubes and cloud-native environments for Earth observation: An overview. Scalable *Computing: Practice and Experience, 25*(6), 5745–5759.

Open Geospatial Consortium. (2023). *OGC cloud optimized GeoTIFF (COG) standard* (Version 1.0). OGC Implementation Standard.

Open Geospatial Consortium. (2022). *Zarr storage specification* (Version 2). OGC Community Standard.

Richards, J. A. (2013). *Remote sensing digital image analysis: An introduction* (5th ed.). Springer.

Roy, D. P., Wulder, M. A., Loveland, T. R., Woodcock, C. E., Allen, R. G., Anderson, M. C., Helder, D., Irons, J. R., Johnson, D. M., Kennedy, R., Scambos, T. A., Schaaf, C. B., Schott, J. R., Sheng, Y., Vermote, E. F., Belward, A. S., Bindschadler, R., Cohen, W. B., Gao, F., Hipple, J. D., … Zhu, Z. (2014). Landsat-8: Science and product vision for terrestrial global change research. *Remote Sensing of Environment, 145*, 154–172.

Selea, T. (2023). AgriSen-COG, a multicountry, multitemporal large-scale Sentinel-2 benchmark dataset for crop mapping using deep learning. *Remote Sensing, 15*(12), Article 2980.

Simões, R., Camara, G., Queiroz, G., Souza, F., Andrade, P. R., Santos, L., Carvalho, A., & Ferreira, K. (2021). Satellite image time series analysis for big Earth observation data. *Remote Sensing, 13*(13), Article 2428.

Schowengerdt, R. A. (2007). *Remote sensing: Models and methods for image processing* (3rd ed.). Academic Press.

SpatioTemporal Asset Catalog. (2021). *STAC specification* (Version 1.0.0).

Sudmanns, M., Tiede, D., Lang, S., Bergstedt, H., Trost, G., Augustin, H., Baraldi, A., & Blaschke, T. (2020). Big Earth data: Disruptive changes in Earth observation data management and analysis? *International Journal of Digital Earth, 13*(7), 832–850.

Sun, Z., Cristea, N., Tong, D., Tullis, J., Chester, Z., & Magill, A. (2023). A review of cyberinfrastructure for machine learning and big data in the geosciences. In *Cyberinfrastructure for machine learning and big data in the geosciences*. Geological Society of America.

Tang, X., Barrett, M. G., Cho, K., Bratley, K. H., Tarrio, K., Zhang, Y., Gu, H., Rasmussen, P., Bosch, M., & Woodcock, C. E. (2024). Broad-area search of new construction using time series analysis of Landsat and Sentinel-2 data. *Science of Remote Sensing, 9*, Article 100138.

Verbesselt, J., Hyndman, R., Newnham, G., & Culvenor, D. (2010). Detecting trend and seasonal changes in satellite image time series. *Remote Sensing of Environment, 114*(1), 106–115.

Witjes, M., Parente, L., Križan, J., Hengl, T., & Antonić, L. (2023). EcoDataCube.eu: Analysis-ready open environmental data cube for Europe. *PeerJ, 11*, Article e15478.

Woodcock, C. E., Allen, R., Anderson, M., et al. (2008). Free access to Landsat imagery. *Science, 320*(5879), 1011.

Wu, S., Song, Y., An, J., Lin, C., & Chen, B. (2024). High-resolution greenspace dynamic data cube from Sentinel-2 satellites over 1028 global major cities. *Scientific Data, 11*, Article 909.

Wulder, M. A., Coops, N. C., Roy, D. P., White, J. C., & Hermosilla, T. (2019). Land cover 2.0. *International Journal of Remote Sensing, 40*(11), 4254–4284.

Xie, Z., Game, E. T., Phinn, S. R., Adams, M. P., Bayarjargal, Y., Pannell, D. J., Purevbaatar, G., Baldangombo, B., Hobbs, R. J., Yao, J., & McDonald-Madden, E. (2024). A scalable big data approach for remotely tracking rangeland conditions. *Communications Earth & Environment, 5*, Article 349.