Efficient Feature Extraction of Petabyte-Scale Datacubes for Weather and Climate

Doctoral thesis
for
the award of the doctoral degree
of the Faculty of Mathematics and Natural Sciences
of the University of Cologne

submitted by
Mathilde Leuridan

in the year 2026

Abstract

Data volumes across scientific domains are increasing rapidly, driven by advances in simulation, observation and data-driven modelling. In meteorology in particular, kilometre-scale numerical weather prediction and high-frequency machine learning models produce vast and continuously growing datasets. While this enables new scientific insights, it also creates fundamental challenges for efficiently accessing and delivering data. A key limitation lies in current data access approaches, which typically retrieve large data subsets from storage systems before applying application-specific transformations, such as clipping. This model is increasingly inefficient due to the mismatch between storage layouts, which are optimised for block-based, partly sequential access on disk or tape, and user-driven access patterns, which are often sparse and require fine-grained random access. At the same time, a growing diversity of users and applications demands more flexible and tailored data extraction capabilities. These challenges motivate the need for fundamentally new extraction paradigms that minimise unnecessary data movement while supporting user-specific queries. This thesis introduces a new feature extraction framework, which shifts data selection logic from the post-processing stage to the data access layer. Instead of retrieving large data blocks and subsequently filtering them, the proposed method computes the exact byte ranges required to satisfy a query in advance, thereby significantly reducing I/O. At its core, the framework introduces a generalised feature extraction mechanism, which first uses a geometric selection algorithm to identify relevant data subsets and then directly retrieves only the necessary bytes from backend storage, enabling highly efficient data access. The proposed approach is implemented in a prototype system and evaluated on operational meteorological datasets. Benchmarks show significant performance improvements compared to existing extraction systems, including reductions in both data transfer volumes, as well as I/O operations, by up to a few orders of magnitude. These results directly translate into lower computational load on backend systems and faster delivery of application-ready data. Beyond regular gridded data, the framework is extended to support unstructured grids without requiring assumptions on spatial layout. Building on this, the approach is further generalised to arbitrary datacubes with a new abstraction that captures complex multi-dimensional data spaces, such as those typically encountered in modern Earth system science. Together, these contributions form a unified and flexible feature extraction mechanism applicable across a wide range of scientific datasets. The presented methods enable efficient, scalable and user-centric data access, particularly in operational environments, such as numerical weather prediction, and digital twin systems, such as the Destination Earth initiative. More broadly, the concepts developed in this thesis are applicable to other data-intensive scientific domains, where minimising data movement and enabling precise on-demand access are critical for future workflows.

Acknowledgements

I would first like to thank my university supervisor, Professor Martin Schultz, for his constant guidance and support throughout my PhD. Over the past years, his advice has helped me navigate the various challenges of this journey. I am in particular grateful for the opportunity to be welcomed into his research group at Forschungszentrum Jülich, where I felt part of an engaging and supportive research community. I greatly enjoyed our many discussions over coffee and appreciated his thoughtful feedback, openness, as well as the trust he placed in my work.

I am equally as grateful to my line manager, Dr James Hawkes, and my section head, Dr Tiago Quintino, who have supported my PhD journey from the very beginning. They introduced me to the world of scientific software development behind modern weather forecasting and encouraged me to explore it in depth. They devoted a great deal of time to my development, patiently discussing ideas, sharing their expertise, and working through problems with me. Their careful reviews and guidance through software design decisions made this experience both challenging and genuinely rewarding.

I am also grateful to ECMWF for providing me with this opportunity and for fostering a welcoming and collaborative work environment. I would in particular like to thank my colleagues, Chris, Adam and Peter, who have helped turn the Polytope software into an operational service over the past few years.

I would also like to thank Professor Stefan Wesner and Dr Carsten Hinz for their valuable comments, expert advice and guidance throughout my PhD, particularly in navigating the University of Cologne and the Jülich Supercomputing Centre.

Finally, I would like to thank my parents and brothers for their constant encouragement and unwavering support. And last but not least, thank you to Sissi and Arthur for their encouraging purrs during the final months of writing.

Publication Preface

This thesis and its contributions are largely based on the following manuscripts.

Published manuscripts

1. Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, Martin Schultz, and Tiago Quintino. "Polytope: an algorithm for efficient feature extraction on hypercubes." Journal of Big Data 12, no. 1 (2025): 1-25. URL 10.1186/s40537-025-01306-3.

2. Mathilde Leuridan, Christopher Bradley, James Hawkes, Tiago Quintino, and Martin Schultz. "Performance Analysis of an Efficient Algorithm for Feature Extraction from Large Scale Meteorological Data Stores." In Proceedings of the Platform for Advanced Scientific Computing Conference, pp. 1-9. 2025. URL 10.1145/3732775.3733573.

3. Mathilde Leuridan, James Hawkes, Tiago Quintino, and Martin Schultz. "An Algorithm for Feature Extraction from Large Scale Meteorological Data Stores on Irregular Grids." Manuscript accepted to the Platform for Advanced Scientific Computing (PASC) 2026.

Preprints or manuscripts under review

4. Mathilde Leuridan, James Hawkes, Tiago Quintino, and Martin Schultz. "Beyond Standard Datacubes: Extracting Features from Irregular and Branching Earth System Data." Manuscript in preparation, 2026.

Non-peer-reviewed manuscripts

5. Mathilde Leuridan, James Hawkes, and Tiago Quintino. "Polytope: Feature Extraction for Improved Access to Petabyte-Scale Earth Observation Datacubes." IAC 2023.

6. Mathilde Leuridan, James Hawkes, and Tiago Quintino. "Polytope: Extracting Features from Large-Scale Datacubes." IAC 2024.

List of Abbreviations

ECMWF	European Centre for Medium-Range Weather Forecasts
NWP	Numerical Weather Prediction
IFS	Integrated Forecasting System
HPC	High-Performance Computing
EO	Earth Observation
OGC	Open Geospatial Consortium
EDR	Environmental Data Retrieval
AI	Artificial Intelligence
ML	Machine Learning

Contents

Abstract v

Acknowledgements vii

Publication Preface ix

List of Abbreviations xi

1 Introduction 1
- 1.1 Traditional NWP Data Systems . . . 2
- 1.2 A New Era of Weather Forecasting . . . 2
- 1.3 Emerging Data and Software Challenges . . . 3
- 1.4 Background and Related Work . . . 5
  - 1.4.1 Modern Data Storage . . . 5
  - 1.4.2 Data Geometry in Earth System Data . . . 7
  - 1.4.3 Data Representations and Datacubes . . . 8
  - 1.4.4 Data Access Tools and Strategies . . . 9
- 1.5 Problem Statement . . . 11
  - 1.5.1 Thesis Outline . . . 12
  - 1.5.2 Contributions . . . 12

2 An Algorithm for Feature Extraction 13
- 2.1 Abstract . . . 14
- 2.2 Introduction . . . 14
- 2.3 Concept . . . 15
  - 2.3.1 Motivation . . . 16
  - 2.3.2 Related Work . . . 16
- 2.4 Polytope Feature Extraction . . . 17
  - 2.4.1 Datacube . . . 17
  - 2.4.2 Slicer . . . 19
- 2.5 Applications . . . 23
  - 2.5.1 Interface . . . 23
  - 2.5.2 Polytope Use Cases . . . 24
- 2.6 Performance and Scalability . . . 26
  - 2.6.1 Performance . . . 27
  - 2.6.2 Bound on Number of Slices . . . 29
  - 2.6.3 Data Reductions . . . 30
  - 2.6.4 Discussion . . . 31
- 2.7 Conclusion . . . 33
- 2.8 Appendix A: Polytope Algorithm Example . . . 33
- 2.9 Acknowledgements . . . 36

2.10 References 37

3 Serving Features on Meteorological Data Stores 39
3.1 Abstract 40
3.2 Introduction 40
3.3 Feature Extraction on Meteorological Data Stores 41
3.3.1 Meteorological Data Stores 41
3.3.2 Polytope 41
3.3.3 GribJump 41
3.3.4 Polytope Tree Compression 42
3.3.5 System Overview 42
3.4 Performance and Scalability of the Polytope System 42
3.4.1 Pre-Computation Phase 42
3.4.2 Online Phase 43
3.5 I/O and Data Reductions 46
3.6 Discussion and Future Work 46
3.7 Acknowledgements 48
3.8 References 48

4 Weather and Climate Digital Twins Use Case 49
4.1 Abstract 50
4.2 Introduction 50
4.3 Polytope Concept 51
4.3.1 Traditional Extraction Methods 51
4.3.2 Polytope Extraction Algorithm 51
4.3.3 Benefits 51
4.4 Weather and Climate Data 52
4.4.1 Datacube Structures 52
4.4.2 Datacube Properties 53
4.5 Polytope in Destination Earth 54
4.5.1 Motivation 54
4.5.2 Supported Features 54
4.5.3 Example 54
4.6 Conclusion 55
4.7 Acknowledgements 55
4.8 References 56

5 Earth Observation Datacubes Use Case 57
5.1 Abstract 58
5.2 Introduction 58
5.3 Data Challenges 59
5.4 Storing EO Data 59
5.4.1 Datacubes 59
5.4.2 Current Initiatives 60
5.5 Polytope Extraction Algorithm 60
5.5.1 Motivation 61
5.5.2 Polytope Mechanism 61

5.6 Example Polytope Use Cases . . . 61
5.6.1 Sea Ice Thickness Example . . . 61
5.6.2 Burnt Land Example . . . 62
5.6.3 Sea Chlorophyll Example . . . 62
5.6.4 Discussion . . . 62
5.7 Conclusion . . . 64
5.8 Acknowledgements . . . 64
5.9 References . . . 64

6 Feature Extraction on Unstructured Grids 67
6.1 Abstract . . . 68
6.2 Introduction . . . 68
6.3 Background . . . 69
6.3.1 Original Polytope Algorithm . . . 69
6.3.2 Irregular Grids . . . 69
6.3.3 Related Work . . . 69
6.4 Extended Slicing Algorithm . . . 71
6.4.1 Modified Algorithm . . . 71
6.4.2 Examples . . . 72
6.4.3 Performance . . . 73
6.5 Discussion . . . 74
6.5.1 Data Extraction as a Service . . . 74
6.5.2 Future Work . . . 76
6.6 Acknowledgements . . . 76
6.7 References . . . 76

7 Beyond Standard Datacubes 79
7.1 Abstract . . . 80
7.2 Introduction . . . 80
7.3 Motivation and Background . . . 82
7.3.1 Datacube Abstraction and Data Representations . . . 82
7.3.2 Datacube Access and Data Extraction . . . 83
7.4 A Datacube Generalisation . . . 84
7.4.1 Data Hypercubes . . . 85
7.4.2 Data Hypercube Operations . . . 86
7.4.3 Data Hypercube Performance . . . 87
7.5 An Integrated Data Extraction System . . . 92
7.5.1 System Architecture . . . 92
7.5.2 System Performance . . . 93
7.5.3 User-Centric Workflows and New Access Patterns . . . 95
7.6 Discussion and Future Work . . . 96
7.7 Acknowledgements . . . 97
7.8 References . . . 97

8 Discussion 103
8.1 A New Data Access Paradigm . . . 103
8.2 Performance Considerations . . . 105

8.3 User-Centric Workflows and Operational Weather Forecasting 107
8.4 Applications Across Scientific Domains 108
8.5 Real-Time and Operational Data Access 112
8.6 Broader Implications 113

9 Conclusion **115**
9.1 Summary of Contributions 115
9.2 Future Work 117

Bibliography **119**

1 Introduction

Weather forecasting plays an important role in modern society, from supporting disaster risk management and improving agricultural planning to predicting future energy demand. Over the past century, improvements in atmospheric science, numerical methods and computing have gradually increased the skill and reliability of forecasts [109, 103, 50]. These advances shape critical socio-economic decisions and have a direct human impact, helping to prevent disasters and enabling early warning systems that allow communities to better prepare for and adapt to environmental change. To meet these needs, state-of-the-art Numerical Weather Prediction (NWP) models, such as the Integrated Forecasting System (IFS) [119] developed by the European Centre for Medium-Range Weather Forecasts (ECMWF), now generate large volumes of data every day. While improving forecast accuracy remains essential, an equally important challenge consists in efficiently delivering model output data to users and downstream applications.

High-performance computing (HPC) and modern software infrastructure have long underpinned weather forecasting and climate science. They not only enable detailed simulations of the Earth system, but also support the large-scale data workflows that happen after each model run [82, 79, 114]. As computational capabilities have advanced, forecasting models have increased in both resolution and complexity. This has led to a steady growth in the volume and diversity of generated data. Meeting these demands depends not only on raw computational performance, but also on software systems that can efficiently store, move and serve data across complex computing environments. Recent developments, such as the European Union's Destination Earth digital twins [118, 42] and the growing use of machine-learning and data-driven forecasting approaches, including systems such as DeepMind's GraphCast [65], further accelerate both data production and data distribution.

Data infrastructure and data management have therefore become central aspects in modern forecasting workflows. As the amount of produced data continues to grow, many existing software technologies and data pipelines are being pushed to their limits. This steady increase in data volume is reshaping not only how forecasts are produced, but also the computational and data management systems needed to support them. Developing scalable and efficient solutions is therefore essential to sustain future progress in weather and climate science.

More generally, these challenges arise in many areas of computer science and data analysis, where increasing data volumes require new methods to access and process large-scale datasets. Such datasets are often described as high-dimensional data spaces, sometimes also referred to as datacubes, which characteristically span multiple coordinate dimensions such as space, time and additional variables. These data spaces can moreover exhibit special properties and other complex behaviours, such as conditional relationships between dimensions. As these data spaces grow, it becomes increasingly difficult to work with them as a whole. Instead, a popular

approach is to focus on accessing specific subsets, or features, which are relevant to a given application. These may correspond, for example, to sub-volumes defined by regular regions or more complex irregular shapes, but can also involve more general query patterns tied to application-specific structures. Supporting these flexible and selective types of data access is therefore becoming an important consideration in the design of scalable data processing systems across a wide range of domains.

1.1 Traditional NWP Data Systems

Modern NWP has its origins in the early 20th century, when researchers first considered solving the governing equations of the atmosphere numerically. This concept became more popular with the emergence of digital computers after World War II, leading to the first operational weather forecast systems. Since then, NWP has developed into a key application of HPC, with continuous advances in computational capabilities enabling higher spatial resolution, more vertical levels, and more sophisticated physical parameterisations.
Operational NWP systems generate a broad set of meteorological variables, such as temperature, pressure, humidity and wind components, as gridded fields defined at the surface and across multiple vertical layers of the atmosphere. These fields can be viewed as individual slices of the atmosphere and are produced at regular forecast intervals to form structured representations of the weather over time. In addition to deterministic forecasts, ensemble prediction systems introduce multiple model runs, called ensemble members, to represent forecast uncertainty and enable probabilistic products.
In traditional operational workflows, these data are handled and distributed as complete spatial fields using standardised formats such as GRIB, as defined by the World Meteorological Organisation. Data access is therefore typically organised around retrieving full variables for a given timestep or specific atmospheric layers, reflecting a field-based view of the atmosphere. Classical forecast products, such as timeseries, surface weather maps, upper-air charts, precipitation accumulations and ensemble-based probability forecasts, are derived directly from these fields in batch-oriented processing workflows. In this sense, weather forecasting has traditionally been approached through the production, distribution, and interpretation of large self-contained data fields, which has shaped both how data are accessed and how forecast products are consumed.

1.2 A New Era of Weather Forecasting

Historically, Numerical Weather Prediction systems produced a limited number of gridded variables at relatively coarse spatial and temporal resolution [85]. Over time, advances in model physics and increasing resolution expanded the number of prognostic and diagnostic fields, while the introduction of ensemble forecasting [45] further multiplied output volumes. As a result, modern NWP systems now generate orders of magnitude more data per forecast run than earlier models [9, 8], turning data management from a secondary concern into a core operational challenge.

In recent years, this trend has accelerated further. Higher-resolution models and the emergence of machine-learning–based forecasting systems have significantly changed how forecasts are produced and how much data they generate. Efforts to run global models at kilometre-scale resolution, such as those in Destination Earth [118, 42, 52, 49], nextGEMS [99, 91] and WarmWorld, contribute substantially to this growth in output size. A single global field at this resolution can already require several gigabytes per parameter and forecast step. With thousands of parameters and around a hundred time steps per run, total forecast outputs can reach hundreds of terabytes. At the same time, data-driven models such as Pangu-Weather [17], GraphCast [65], FourCastNet [90] and AIFS [66] demonstrate that forecasts can now be generated at unprecedented speed. These systems are capable of producing large volumes of data much faster than traditional physics-based models, further increasing demands on data handling and distribution. While the extent to which this capability translates into operational requirements depends on the application, with short-term forecasting benefiting from more frequent updates that help capture rapidly evolving conditions and medium-range predictions generally requiring them less, these developments highlight the potential to generate forecasts at an entirely new scale and speed.

NWP output is also increasingly accessed alongside other data sources, including Earth observations, climate projections and, potentially, high-frequency sensor and IoT data. As a result, weather and Earth system science now exhibit the defining characteristics of *big data*, which are high volume, high velocity and high variety [35, 26, 101]. This marks a significant shift for a field that, for many years, operated with relatively stable data workflows.

In parallel, the growing adoption of digital twins in Earth science introduces a second major change. Projects, such as the European Union's Destination Earth initiative [118, 42] or the various Digital Twin of Earth [20, 7] models, promote interactive systems in which data can be explored, combined and accessed dynamically. These systems place users at the centre of the workflow and require infrastructure that supports flexible, low-latency access to heterogeneous datasets. This differs from the traditional paradigm in which forecasting systems primarily produced data in fixed formats. In that model, most processing and data transformation were left to the user. Modern systems instead aim to support user-centric workflows. Data should be easy to access, understand and combine in different ways, depending on the task. Supporting such workflows therefore requires a fundamental change of the software technologies used in operational weather forecasting.

1.3 Emerging Data and Software Challenges

With the shift toward an era of big data, weather forecasting systems now face challenges that were largely absent in earlier generations of NWP. A key factor underlying these challenges is the nature of meteorological data itself, as well as the operational context in which it is produced and used. Weather data is inherently time-sensitive and its value rapidly decreases, whether it is used for disaster prevention, which requires timely updates during unfolding events, or for energy demand forecasting, which depends on fresh data to guide operational decisions. As a result,

weather forecasting systems operate under strict time constraints to deliver data reliably to users. Although this time sensitivity has always been a fundamental characteristic of meteorological data, as data volumes and processing complexity grow, meeting these timeliness requirements becomes increasingly difficult in practice.

At the same time, these meteorological data systems need to accommodate a user base that is growing in size and diversity. In addition to traditional users such as forecasters, a wider range of groups, including for example emergency and rescue services as well as energy infrastructure operators, now require more direct access to NWP data. Many users may access the same data simultaneously, often with strict requirements on availability, robustness and performance. Systems must therefore be both efficient and highly scalable. The rapid increase in the volume and velocity of NWP output further increases the pressure on these systems, mainly by placing greater technical demands on data processing, transfer and overall performance. Existing workflows face increasing difficulty to meet growing user demand for timely and efficient access to data. This marks a point where traditional approaches begin to reach their limits.

An additional challenge arises from the growing expectation that NWP systems should better support user-centric workflows. Data should be accessible not only to domain experts, but also to users with limited background knowledge and to downstream applications. However, NWP data is inherently high-dimensional and complex, spanning many variables, spatial and temporal scales, as well as data sources. This complexity makes it difficult for users to understand what data is available and how best to access it. From a service perspective, user-centric workflows also translate into a wide range of access patterns driven by diverse use cases, rather than purely by the underlying degrees of freedom of the data. In particular, this introduces several ways of querying and interacting with the data, which complicates both exploration and efficient access. Designing systems that can accommodate these varied access patterns while maintaining performance and scalability therefore becomes a central technical challenge. Indeed, supporting intuitive and flexible workflows represents a significant departure from earlier systems, which prioritised data production over usability. In particular, it requires rethinking system architectures to better balance both efficiency and flexibility in the context of evolving user demands.

Together, these developments imply a need to rethink the design of data systems and workflows in weather forecasting. A few core challenges emerge from this shift. First and foremost, there is an increasing need for more efficient and user-oriented data retrieval. Users are rarely interested in downloading entire global fields. Instead, they usually want answers to specific questions, such as temperature values along a timeseries or within a given region. Flexible multidimensional data retrieval directly addresses this need by enabling access to only the data relevant to a given request. By restricting queries to the specific spatial, temporal and variable subsets that are really needed, the system avoids reading and processing large volumes of irrelevant data. This leads to faster query execution, lower backend I/O, as well as better scalability when serving many concurrent users. Overall, accessing only what is needed enhances performance while also minimising wasted computation

and storage overhead.
Second, there is a need for accurate and usable representations of the data space itself. Given the high dimensionality and the complex heteorogenity of NWP data, users require clear and consistent ways to explore what data is available and how it can be accessed. This is essential for designing reliable downstream workflows and applications. A well-defined and explorable data representation also benefits the systems themselves, as it enables more informed optimisation, as well as more robust data handling. Data exploration is therefore a core aspect of any system that aims to be both data-centric and user-centric.
Addressing these two challenges, efficient data retrieval and insightful data exploration, is essential to build future meteorological data systems that are robust, scalable and able to meet the evolving demands of modern forecasting workflows.

1.4 Background and Related Work

Although the scale and complexity of these challenges continue to grow, many of them have long been present in computer science. Consequently, the community has developed a variety of approaches to address them, covering both data exploration and data extraction across areas such as database systems, high-performance computing and distributed data processing. Similar challenges have also arisen in other scientific domains, where alternative concepts and solutions have been proposed. Some of these ideas have since been adapted to Earth science and have started to shape the development of more robust and efficient data systems, while others prove more difficult to transfer due to domain-specific characteristics of Earth system data. However, many existing approaches remain limited in their ability to cope with increasing data volumes, evolving data patterns and growing demands for flexible, user-driven data access.
To provide the necessary foundation for this discussion, the following section first introduces key concepts of modern data storage and fundamental aspects of data geometry, including those particularly relevant to Earth system data. Building on this foundation, we then review key approaches that have emerged in response to the challenges of designing robust and efficient data systems in the era of big data, and discuss how they have influenced current practice in NWP. In particular, we examine commonly used data representations in Earth science, along with widely adopted data access tools and strategies. This review provides the foundation for identifying remaining gaps and motivates the need for more scalable, flexible and user-oriented solutions, which are the focus of this thesis.

1.4.1 Modern Data Storage

Modern data management systems are built around a range of storage technologies, which differ in their capacity, performance and cost characteristics. These technologies are usually not used in isolation. Instead, they are combined into layered systems [61, 121], which span from long-term archival storage to high-performance devices, allowing both persistent storage and efficient access when needed.
At the foundation of this hierarchy are established technologies, such as spinning

disks [102, 78] and tape storage [67, 58, 56]. Hard disks are still widely used as they offer large storage capacity at a relatively low cost, with typical access times on the order of a few milliseconds, and sequential read speeds reaching several hundred MB per second. Tape systems follow the same idea but are aimed at much larger-scale storage. They provide extremely low cost per byte and are well suited for storing data at the petabyte scale. These systems are however designed for sequential access and retrieving data can take anywhere from seconds to minutes. They are therefore mainly used for long-term archiving rather than for interactive work. These are complemented by faster solid-state storage [96, 2], such as NVMe devices [120], which offer much lower latencies, normally on the order of tens of microseconds, and significantly higher throughput, often reaching several GB per second per device. They are often used when data needs to be accessed quickly and repeatedly, such as during intermediate processing steps or in performance-critical parts of a workflow for example.

A key distinction across all storage systems is between sequential and random access [3, 126, 60, 16]. Sequential access refers to reading data in contiguous blocks and usually allows for high throughput, whereas random access involves retrieving many small pieces of data from different locations, which introduces overhead and can slow operations down significantly, especially on systems with mechanical components. This is why the way data is laid out and accessed plays an important role in overall performance. Data is transferred between storage and memory in units of a defined block length. Typical block sizes range from a few kilobytes to several megabytes, depending on the system. This implies that even small data requests may involve reading larger chunks of data. Modern systems reduce repeated access overhead through buffering and caching [29, 127], keeping frequently accessed data in faster memory such as RAM. Cache sizes can range from megabytes in CPU caches, to many gigabytes in system memory, and can significantly influence effective performance of data systems.

At larger scales, storage is commonly organised through distributed and parallel file systems in HPC environments [81, 77, 18]. These systems distribute data across multiple storage nodes and enable concurrent access by many processes, achieving aggregate bandwidths of tens to hundreds of GB per second. They are designed to support large tightly coupled workloads and are widely used in scientific computing. Alternatively, cloud-based storage has become increasingly common [94, 97, 47], using object storage to abstract away the underlying hardware and support scalable access to large datasets. Whilst cloud storage offers considerable flexibility and ease of use, access latencies are often higher, commonly on the order of tens to hundreds of milliseconds, and throughput depends on network performance and access patterns. As a result, efficient use of cloud storage often involves structuring data and requests to minimise overhead and maximise data transfer efficiency.

All in all, these technologies illustrate that modern storage systems are not defined by a single performance profile, but rather by a range of different capabilities. Small datasets can often be handled with simple access patterns and minimal concern for storage layout. In contrast, at the scale of terabytes and petabytes, differences in latency, bandwidth and access patterns become decisive. Understanding these characteristics provides the foundation for interpreting data processing performance and

motivates the need for tailored strategies in large-scale data environments.

1.4.2 Data Geometry in Earth System Data

Besides storage systems, the data structure itself also plays an essential role in how it can be accessed and processed. In many cases, the data geometry determines which operations are easy or difficult to implement, and this becomes especially relevant to consider for large datasets, where even simple tasks can be affected by more complex data structures.
An important factor to take into account is the difference between structured and unstructured grids [107, 108]. Structured grids are defined through mathematical relationships. As a result, the exact position of each data point can be computed directly from its grid index. In Earth system sciences, these are often iso-latitude grids, where spatial coordinates follow a consistent pattern. This explicit structure simplifies both data representation and data access. Unstructured grids do not follow such a fixed pattern, and instead of being derived from a simple formula, the location of each data point has to be described explicitly. Whilst this enables flexible representations of complex spatial domains, it also introduces additional complexity in data handling.
Earth system datasets, and NWP datasets in particular, are inherently multidimensional. In addition to spatial dimensions, they include multiple temporal axes and often further dimensions such as ensemble members, which represent different realisations of a forecast to capture uncertainty [76]. A key feature is the presence of two distinct time dimensions, the forecast reference time, which indicates when a forecast was generated, and the forecast lead time, which describes the time progression within the forecast itself. Another important property of these datasets is that some spatial axes can be cyclic, such as longitude for example, which introduces wrap-around behaviour at domain boundaries and must be explicitly accounted for in indexing, interpolation and neighbourhood-based operations.
These structural characteristics are reflected in widely used data formats, such as NetCDF [95] and GRIB [15, 113], which provide standardised ways of representing multidimensional geoscientific data. In NetCDF, data is organised along named dimensions and variables, making it easy to understand the relationships between different axes. On the other hand, the GRIB format encodes data together with metadata in compact messages, where information about dimensions, grid structure and forecast context is defined through standardised headers. In both cases, strict conventions govern how data axes and associated metadata are defined, ensuring interoperability but also shaping how data can be accessed and interpreted.
Together, these aspects illustrate that the organisation of Earth system data is not arbitrary, but follows well-established structural and format-specific conventions. Understanding these conventions provides the necessary context for examining how different data representations and access strategies have evolved to address the challenges of large-scale data processing.

1.4.3 Data Representations and Datacubes

High-dimensional data is common across many scientific disciplines. Because of its inherent complexity, researchers have developed structured abstractions that provide simplified and consistent views of the data, allowing it to be explored, analysed, and reasoned about more effectively. These abstractions play a central role in enabling efficient data exploration and have strongly influenced the design of software systems and analysis tools for large-scale scientific datasets.

A significant body of work in this area has focused on defining orthogonal, tensor-based views of multi-dimensional data [68, 33, 25]. By organising data along clearly defined and independent axes, such as space, time and parameters, these representations allow users to apply consistent operations across dimensions, including slicing, aggregation, and subsetting. Such tensor views have proven particularly effective in the field of Earth science [124, 12], where many datasets naturally conform to structured high-dimensional grids.

One of the earliest and most popular implementation of this idea is the datacube abstraction [48, 64], originally developed in the context of business analytics for tabular data [84]. This concept has since been refined and widely adopted in Earth science, especially for Earth Observation (EO) data [111, 14, 13, 44]. EO datasets typically share a small number of dominant dimensions, most commonly latitude, longitude and time, over which a range of physical variables, derived products or instrument measurements are defined. These dimensions provide a convenient and intuitive basis for organising data into cubes and for expressing common scientific queries.

This abstraction underlies many prominent EO datacube initiatives, including global systems such as the DeepESDL datacube [4, 27] and numerous national or regional platforms, such as the Swiss [43], Austrian [112] and Brazilian [37] datacube frameworks. Collectively, these efforts have demonstrated the practical value of a common data model. They have in particular enabled interoperability between systems, facilitated software reuse and accelerated the development of analysis workflows.

However, these achievements rely on a set of strong assumptions about data structure and dimensional regularity. Most existing data cubes impose fixed grid-aligned dimensions and require data to be laid out in a predefined and uniform manner. Whilst this rigidity simplifies many operations, it becomes increasingly restrictive as data volumes grow and data variety expands, especially in meteorological and environmental modelling contexts where grids, resolutions or domains may vary between datasets or over time. Furthermore, the datacube remains primarily a conceptual abstraction for viewing and manipulating data, which does not directly address how data is stored, indexed or accessed in backend systems. In practice, implementing this abstraction at scale is non-trivial. Indeed, while a datacube can simply be viewed as a multi-dimensional array in principle, real-world constraints such as memory limits, storage layouts and file system structures require additional engineering to make large datasets appear and behave like a coherent datacube. This introduces a separation between the logical datacube representation and the underlying physical storage layer. As a result, efficient and scalable data access depends on deeper integration with data management and storage tools, highlighting limitations in current approaches and motivating the need for more flexible solutions.

1.4.4 Data Access Tools and Strategies

Beyond defining uniform abstractions of data spaces, substantial effort in Earth science has focused on standardising and improving how data is accessed and extracted. In particular, a range of community standards and interfaces have emerged to formalise common data access patterns. Notably, the Open Geospatial Consortium (OGC) has introduced feature-based concepts and associated interfaces, such as the Environmental Data Retrieval (EDR) API [22], which allow users to request specific spatio-temporal subsets of data, such as points, trajectories and polygons. These standards aim to make data access more user-oriented by enabling queries that return only the data relevant to a specific scientific question, rather than entire datasets. While conceptually powerful, these feature-based approaches are still relatively new, and efficient, scalable implementations do not yet exist. In parallel, a wide range of tools and backend strategies have been developed to support efficient access to large, multi-dimensional datasets [41, 98, 38]. These systems operate not only on metadata, but directly on underlying storage in order to execute complex operations and enable performant workflows. Their design is therefore tightly coupled to data formats and storage layouts. In recent years, software packages such as xarray [53] and Zarr [123, 83] have gained significant traction in the Earth science community. Xarray provides a high-level, tensor-based view of multi-dimensional data tailored to climate and weather applications, and offers native support for common meteorological formats such as NetCDF, GRIB, and HDF [28]. Zarr, by contrast, has emerged as both a storage format and access framework for large datasets, primarily due to its support for chunked storage.

Chunking has become a popular strategy for scalable data access in weather forecasting and Earth sciences more generally, as model outputs are usually too large to load into memory as a whole. By partitioning data into smaller, independently retrievable blocks, systems can load only the portions required to answer a given query. This approach is now fundamental to many big data workflows and underpins many modern systems such as Rasdaman [11, 10], SciDB [110, 21] and ClimateSpark [54], which reorganise datasets into carefully designed chunk layouts rather than storing them in their original full form, as summarised in Table 1.1. Chunking therefore significantly improves both performance and scalability by reducing unnecessary data transfer, as well as expensive data transformations.

However, chunking also introduces a key limitation as performance becomes highly sensitive to the chosen chunk layout. Because chunking implicitly optimises data access along specific dimensions, it favours certain query patterns over others [87]. As datasets grow in size, dimensionality and structural diversity, it becomes increasingly difficult to define a single chunking strategy that performs well across all access patterns. In practice, this means that systems are often optimised for a specific class of queries, while others may incur substantial unnecessary data access and degraded performance. Complementing chunk-based access, feature extraction and spatial subsetting [100, 117, 93, 51] have long played an important role in Earth science data systems, as many users are primarily interested in localised regions or specific geometric features rather than global fields. Systems such as RasDaMan support this by identifying and loading only those chunks that intersect a requested feature, while spatial databases like PostGIS [89, 125] provide clipping operations,

Table 1.1: Comparison of chunk layout strategies and typical axis-oriented chunking in array-based systems for climate and weather data

System	Data Model	Chunk Layout Strategies	Typical Axis-Oriented Chunking (Climate Data)
Rasdaman	n-dimensional array (datacube)	Regular and directional tiling, query-driven tiling	Chunking is adapted to access patterns. Common layouts include spatial tiles (latitude $\times$ longitude) for map queries and time slabs for timeseries extractions.
SciDB	n-dimensional array with attributes	Regular chunks with user-defined size and shape	Uses fixed chunks aligned with dimensions (time, latitude, longitude, level). Common setups include small time depth with larger spatial regions, or balanced chunks when dimensions are similar, tuned in particular for distributed processing.
ClimateSpark	n-dimensional array abstraction over NetCDF and HDF	Chunking inherited from the underlying data formats	Inherits NetCDF/HDF chunking, usually per parameter. Common access patterns include time-aligned chunks for temporal analysis and spatial tiles with shallow time depth. Dimension order (often time-first) affects access performance.

typically applied as a post-processing step after data has been accessed. While these approaches improve usability, feature extraction is in most cases treated as an operation layered on top of fixed chunk layouts. As a result, feature-based queries remain constrained by storage decisions made at data creation time and cannot fully adapt data access to the shape, orientation, or dimensionality of the requested feature. This separation between high-level feature-based query semantics and low-level data access strategies represents a key limitation of many modern systems and further motivates the need for approaches that more closely align user-driven queries with backend data access.

1.5 Problem Statement

The following work investigates a feature extraction approach in which regions of interest are handled directly at data access time, rather than as a post-processing clipping operation. Instead of reading larger data blocks and subsequently discarding unwanted portions, we compute the exact data locations and byte ranges required to satisfy a user-defined query, and access only those bytes from the backend. To enable this, we adopt a shape-agnostic model based on arbitrary polytopes in high-dimensional data spaces, rather than relying on predefined query templates or fixed extraction shapes. This allows feature extraction to be expressed directly in the native coordinate space of the data and makes the approach applicable across a wide range of scientific domains. Domain- or standard-specific interfaces can then be implemented on top of this core mechanism without changing the underlying extraction logic.

A key distinction from existing systems is that feature extraction is not implemented as a filtering or clipping step after data retrieval, nor does it rely on chunk-based access patterns. By computing which exact parts of the dataset intersect the region of interest, our approach avoids reading surrounding data and instead accesses only the true data of interest. In many cases, this directly results in improved efficiency, scalability and robustness of the data extraction system, in particular for complex queries and large datasets. This is achieved by reducing unnecessary data transfer and post-processing overhead. The exact performance, however, depends on the characteristics of the underlying storage system, as well as on the access pattern of specific queries. Many backends, and in particular disk-based systems, operate on block-level reads. This implies that data is retrieved in contiguous segments regardless of the requested subset. As a result, highly fragmented access patterns or systems with limited random access performance may reduce these performance benefits, while more sequential queries can still favour chunk-based access strategies. It is thus important to consider both the system architecture and the query structure when evaluating the overall efficiency of this feature extraction method.

Besides these performance considerations, the generic nature of the approach supports its application to a wide range of data representations. In particular, this new method naturally extends to more complex data spaces, including those with unstructured grids, branching coordinate dimensions and non-trivial data constraints. It therefore provides a foundation for a fully generic and adaptable data extraction framework.

1.5.1 Thesis Outline

The thesis begins by introducing Polytope, a flexible feature extraction algorithm, in Chapter 2. This algorithm identifies coordinate points contained within arbitrary n-dimensional polytopes when the underlying data is well aligned in space. This offers an efficient alternative to traditional clipping operations. Chapter 3 then shows how this algorithm can be integrated into an operational weather forecasting environment, enabling an end-to-end feature extraction framework for climate and weather data. Chapters 4 and 5 present concrete applications of the approach on the Destination Earth dataset and on other more generic Earth observation datasets respectively. In Chapter 6, the original algorithm is further extended with a second method capable of handling more complex data spaces, focusing in particular on unstructured grids. Finally, Chapter 7 introduces a new generic representation of complex datacubes, which enables virtually any scientific data space to be modelled. This representation is then integrated into the proposed feature extraction framework to form a fully generic and adaptable data extraction system with minimal assumptions about the underlying dataset. To conclude, the final discussion looks at the broader implications of this work. In particular, it examines the new types of scientific data access workflows which this approach enables, as well as its potential applications across a wide range of scientific use cases.

1.5.2 Contributions

This thesis makes the following core contributions:

- The introduction of a shape-agnostic feature extraction algorithm based on computational geometry concepts, enabling broad applicability across scientific domains.
- A departure from chunk-based data access in favour of precise, region-aware extraction, significantly improving efficiency and scalability.
- The design and implementation of Polytope as a reusable, extensible algorithm for feature extraction in both structured and unstructured data spaces.
- A new generic datacube representation which supports complex grids and data constraints, forming the basis of a fully adaptable extraction framework.

Together, these contributions form the basis for a unified and user-oriented approach to feature extraction that is efficient, flexible and capable of operating across a wide range of datasets and application domains.

2 An Algorithm for Feature Extraction

Traditional data extraction methods in weather forecasting typically rely on orthogonal, box-shaped queries over multidimensional datacubes. In practice, however, many users require data corresponding to more complex geometries, such as country cut-outs or other geographical regions of interest. Advanced use cases can even include the extraction of data along trajectories, for example along flight paths or shipping routes. These use cases are poorly served by box queries, in which we extract the full bounding box of the region of interest and which therefore often result in substantially more data being retrieved than required.
Existing approaches address this limitation through post-processing, by extracting the desired shape after the full bounding box has been retrieved. However, this remains inefficient, as the unnecessary data has already been read. In the following paper [74], we introduce an alternative approach through the Polytope algorithm. The algorithm determines, prior to extraction, the exact data bytes required, enabling direct retrieval of arbitrarily shaped regions. We describe the algorithm for extracting arbitrary high-dimensional polytopes from xarray DataArray objects and demonstrate several applications in meteorology and healthcare. Finally, we evaluate its performance and scalability for high-dimensional shapes and large-scale datasets.

This work was published in Springer's Journal of Big Data, and the Polytope source code [75] is publicly available on GitHub. I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *J. Hawkes*: Conceptualisation, Software, Writing-review & editing; *S. Smart*: Conceptualisation; *E. Danovaro*: Conceptualisation; *M. Schultz*: Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

Leuridan *et al. Journal of Big Data* (2025) 12:243
https://doi.org/10.1186/s40537-025-01306-3

Journal of Big Data

RESEARCH Open Access

Polytope: an algorithm for efficient feature extraction on hypercubes

Mathilde Leuridan[1,2*], James Hawkes[1], Simon Smart[1], Emanuele Danovaro[1], Martin Schultz[2,3] and Tiago Quintino[1]

*Correspondence:
Mathilde Leuridan
mathilde.leuridan@ecmwf.int
[1]European Centre for Medium-Range Weather Forecasts (ECMWF), Reading, United Kingdom
[2]Department of Mathematics and Computer Science, University of Cologne, Cologne, Germany
[3]Jülich Supercomputing Centre, Forschungszentrum Jülich (FZJ), Jülich, Germany

Abstract
Data extraction algorithms on data hypercubes, or datacubes, are traditionally only capable of cutting boxes of data along the datacube axes. For many use cases however, this returns much more data than users actually need, leading to an unnecessary consumption of I/O resources. In this paper, we propose an alternative feature extraction technique, which carefully computes the indices of data points contained within user-requested shapes. This enables data storage systems to only read and return bytes useful to user applications from the datacube. Our main algorithm is based on high-dimensional computational geometry concepts and operates by successively reducing polytopes down to the points contained within them. We analyse this algorithm in detail before providing results about its performance and scalability. In particular, we show it is possible to achieve data reductions of up to 99% using this algorithm instead of current state of practice data extraction methods, such as meteorological field extractions from ECMWF's FDB data store, where feature shapes are extracted a posteriori as a post-processing step. As we discuss later on, this novel extraction method will considerably help scale access to large petabyte size data hypercubes in a variety of scientific fields.

Keywords Data management, Data processing, Data extraction, Datacube, Computational geometry

Introduction

In the past century, fields in science and technology have entered a new era - the era of "big data" [1]. From weather forecasting to astronomy and medicine, scientific advances have led to a surge in the quantity of data produced daily. Indeed, scientific data has been steadily growing in the past decades and in recent years especially, it has experienced exponential growth [2, 3]. y promises for major scientific developments in the years to come, the question arises of how to efficiently use this wealth of data.

The scientific data collected nowadays often depends on a number of different variables and can thus be represented as a multidimensional array, or datacube [4]. Organising data inside such datacubes has attracted a lot of interest in the past few years, with many tools now available to work on such data representations [5–8]. Most modern software architectures provide support to handle such data structures, from cells in

Matlab [9] to Xarray [10] in Python and xtensor [11] in C++. However, in each of these tools and software, datacubes can only be accessed orthogonally to their axes by selecting specific values or ranges along given dimensions. This can most easily be seen in datacubes which use SQL-type queries through the WHERE keyword [12–15], but it also holds true more generally for other datacube implementations which only support axis-aligned subsetting operations on the data [16–19].

Such limited data access mechanisms in the form of bounding boxes are sub-optimal for a wide range of applications. Consider for example the case where a user wants to access temperature data over a country. For this particular example, the bounding box data extraction approach proves to be quite inconvenient as country shapes are not well-represented by bounding boxes. Alternatively, consider a user who wants to access data over some spatio-temporal path [20]. If the entire bounding box of data around the requested path is extracted, the user then has to mask out the wanted path in order to find the data points relevant to his application [21]. This thus not only implies that much more data than is necessary is read and returned from the datacube, thereby consuming more I/O resources, but it also places the additional burden of post-processing on the user after retrieval.

To address this issue, we introduce a new alternative way of accessing datacubes. Our extraction algorithm, Polytope[1] [22], enables users to efficiently query arbitrary high-dimensional shapes from a datacube, slicing non-orthogonally at arbitrary angles against the datacube's axes.

As shown with the country slicing example, traditional bounding-box extraction methods are insufficient for handling complicated non-rectangular requests. The Polytope algorithm however was designed especially with these requests in mind and is able to directly extract such shapes from very large data hypercubes. Because our algorithm computes the exact data points that users are interested in and only reads those from the datacube, it scales well to large high-dimensional request shapes unlike the traditional bounding-box extraction techniques which scale with the tensor product of each dimension. The Polytope algorithm will thus enable scientists to efficiently make use of their ever increasing data, whilst improving the efficiency of their I/O system.

The Polytope algorithm will be used as part of the Polytope service in both the ECMWF Software EnginE (ESEE), which facilitates all data provision and workflow management services at the European Center for Medium-Range Weather Forecasts (ECMWF), and the Destination Earth (DestinE) Digital Twin Engine (DTE) [23].

In this paper, we first introduce the idea behind the Polytope algorithm, before describing its inner mechanism in detail. We then expose some of its possible applications in different scientific fields before finally performing an analysis of the algorithm's performance on selected test cases.

Concept

Before diving into a technical description of the algorithm, let us first explain in more detail the conceptual approach we take. With the Polytope algorithm, we developed a data extraction algorithm which supports the retrieval of arbitrary high-dimensional request shapes, called *features*, from arbitrary multi-dimensional, high-volume data

[1] https://github.com/ecmwf/polytope

hypercubes. Our algorithm is not restricted to any particular request shapes or application field and is in fact intended to be generic and work seamlessly in any scientific application involving multi-dimensional datacubes.

Motivation

Rather than pre-defining a set of shapes which can be extracted from the datacube, the Polytope algorithm takes n-dimensional *polytope* shapes as input, giving the algorithm its name. In computational geometry, a polytope (or bounded convex polytope) is defined as the convex hull of a given point set $\mathcal{P} = \{p_1, \ldots, p_n\}$ [24, 25].

Note that polytopes are convex by definition. Polytopes can in fact be thought of as multi-dimensional convex polygons.

In the Polytope software, we use polytopes because any arbitrary high-dimensional shape, even a concave shape, can be either approximated by or decomposed into simpler convex polytopes [26]. Indeed, polytopes can be seen as the building blocks of high-dimensional geometry. They form the basis of most modern meshing softwares and are widely used in computer graphics to model intricate objects [27]. We thus see that by formulating data requests as polytopes, users will in theory be able to request almost any feature of interest to them from a datacube.

The underlying idea behind the Polytope algorithm can be visualised in Fig. 1.

Related work

There are many alternative data extraction methods based on spatial indices present in literature. Similarly to the solution we propose here, some of these methods were especially designed to extract polytopes, and 2-dimensional polygon shapes in particular, from spatial datasets.

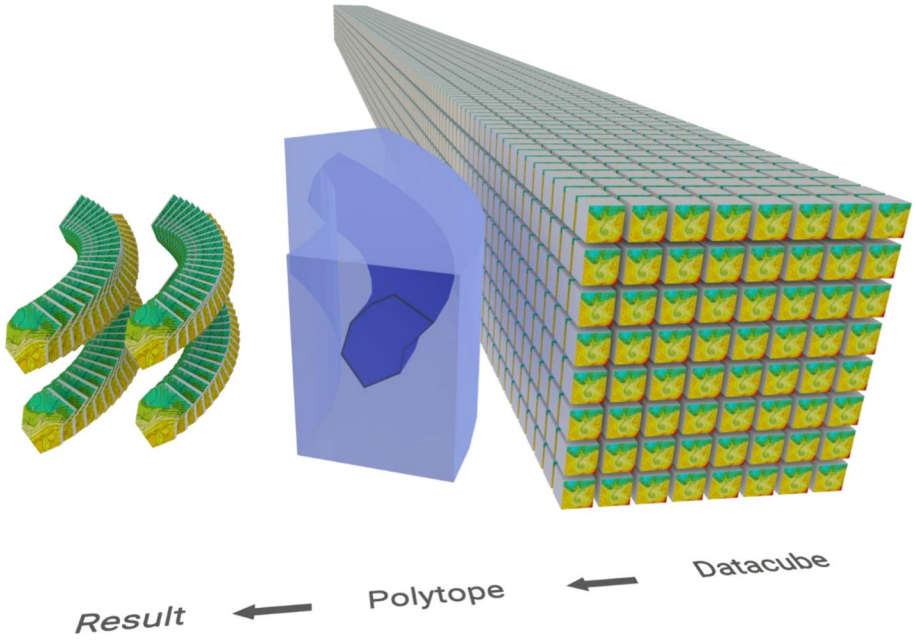


Fig. 1 Concept of the Polytope algorithm. A 6-dimensional polytope stencil is used to extract data of this shape from the datacube. In dark blue, we picture the innermost 3-dimensional shape to extract, which is extruded into a 6-dimensional shape by taking its tensor product with the light blue shape over the outer 3-dimensional datacube

One popular such approach is to build R-trees [28] of 2- or 3-dimensional datasets to allow for efficient querying of an area. This approach already constitutes an important improvement to more traditional bounding box methods where the entire bounding box of the area is extracted, as it allows the system to extract a smaller bounding box coverage of the requested polygon. For very large high-resolution datasets however, this method might still return many more data points than needed and would require expensive point-in-polygon tests to further refine the output to only the points of interest. In particular, this method strongly depends on the constructed R-tree and its depth and resolution. As the R-tree is a static object for each dataset, the initial choice of how it is constructed will therefore significantly affect the subsequent extraction speeds of different areas. For example, when extracting smaller polygons, a larger R-tree with smaller leaf bounding boxes would return data closer to the user request, but having such a larger tree would also make extraction more costly for larger polygons. Our proposed Polytope algorithm does not require building such a static object and the extraction thus remains completely dynamic regardless of the requested shape. Furthermore, as the proposed Polytope algorithm only returns the points of interest, no post-processing masking operation is needed either.

Another alternative method of interest here is the RasDaMan [15] approach. RasDaMan first subdivides datasets into rectangular tiles to then be able to read those independently and perform queries on them once loaded in memory. Similarly to R-trees, this method implies that a rectangular coverage of the shape of interest is accessed. The main aim of the Polytope algorithm is to prevent such unnecessary data access and to instead pre-compute the points of interest in a file before accessing them, which results in a very different extraction method. Furthermore, RasDaMan and other similar methods, such as the clipping operation available in PostGIS [29], only support 2-dimensional polygon masking as a post-processing operation. In comparison, the Polytope method described in this paper has many more degrees of freedom and allows the extraction of arbitrary polytope shapes in any dimension.

Polytope extraction algorithm

We now introduce the Polytope feature extraction algorithm, highlighting in particular the way in which it achieves polytope-based feature extraction on datacubes.

Note that, even though the Polytope algorithm is quite generic, the datacubes on which it operates need to possess some particular properties. We thus first discuss some of these datacube properties before describing the complete mechanism behind the feature extraction algorithm.

Datacube

Datacubes can be thought of as multi-dimensional arrays. In particular, they store data points along different datacube dimensions [30]. Each datacube dimension has an associated coordinate metadata "axis" with a discrete set of indices stored on it [31]. A data point is then located at each of these indices, forming a datacube. Datacubes can store structured point clouds or raster data for example.

A datacube can have different properties depending on the data that it stores. It can for example exhibit splitting behaviour when two datasets are incompatible, be unbalanced when some experiments gather more data than others or even have gaps between data points if results are not regularly recorded. In this section, we identify and describe these properties, as well as other essential datacube features, such as the datacube axes.

This helps us construct a common framework for treating various types of datacubes throughout our algorithm.

Axes

Axes in a datacube refer to the coordinate dimensions along which the data is stored. Values along these axes are called indices. In the following, we differentiate between two main types of axes, the ordered and unordered categorical axes. These two types of axes cannot be treated in the same way within the slicing step of the algorithm, which leads to their distinction here.

Ordered Axes These axes only accept sets of comparable indices which can be ordered. In particular here, this means that values on ordered axes need to be comparable to each other, such that they must meaningfully support comparison operators ($==$, $<$, $\leq$, $>$, $\geq$). This property then directly implies an ordering between indices on ordered axes. Importantly, note that indices on ordered axes do not have to be integers, but can in fact be any countable type that supports a comparison operation, such as time entities, floating point numbers and of course integers. For such axes, it is possible to query ranges of indices as well as individual axis values.

Categorical Axes The other type of axes which can be handled by our algorithm are categorical axes. These axes only support distinct indices which are not comparable to each other, such as string indices for example. In this case, unlike for ordered axes, it does not make sense to query ranges of indices. Instead, the only possible queries on categorical axes are specific index selections.

Note that, in practice, indices on a datacube will always be discrete as there will always be a gap between values of digitally stored data. All ordered axes are thus countable axes, for which indices can be ordered and numbered using natural numbers [32]. Note also that the indices on ordered axes do not have to be uniformly spaced. In particular, the datacube axes can be irregular and sparse in their indices [33]. Lastly, observe that ordered axes can exhibit special behaviours, such as cyclicity along their indices. These behaviours are applied as post-processing operations on the axis indices and therefore do not modify the ordering of the axes.

All categorical axes can be treated in the same way. Similarly, all ordered axes can also be treated in the same fashion. This allows us to take a common approach towards extracting indices on those axes and thus facilitates the data extraction algorithm.

Datacube structure

A datacube can be viewed as a possibly non-regular imbalanced tree. This can be seen in Fig. 2 and we now explain each of these two datacube properties, non-regularity and imbalance, in more detail with the help of an example.

Note that the datacube does not necessarily have the same dimensionality in all directions. On some axes, it is possible to have axis indices which give rise to different subsequent axes or axis values. Consider for example the datacube in Fig. 2 which has indices **val4** and **val5** on its **ax2** axis. If we pick index **val5**, the leaf datacube is 2-dimensional with axes u and v, whereas if we pick index **val4** instead, the leaf datacube is 3-dimensional with axes x, y and z. This phenomenon can be viewed as a non-regular branching of the datacube axes. This is an important feature of the datacube, which we should take into consideration when thinking

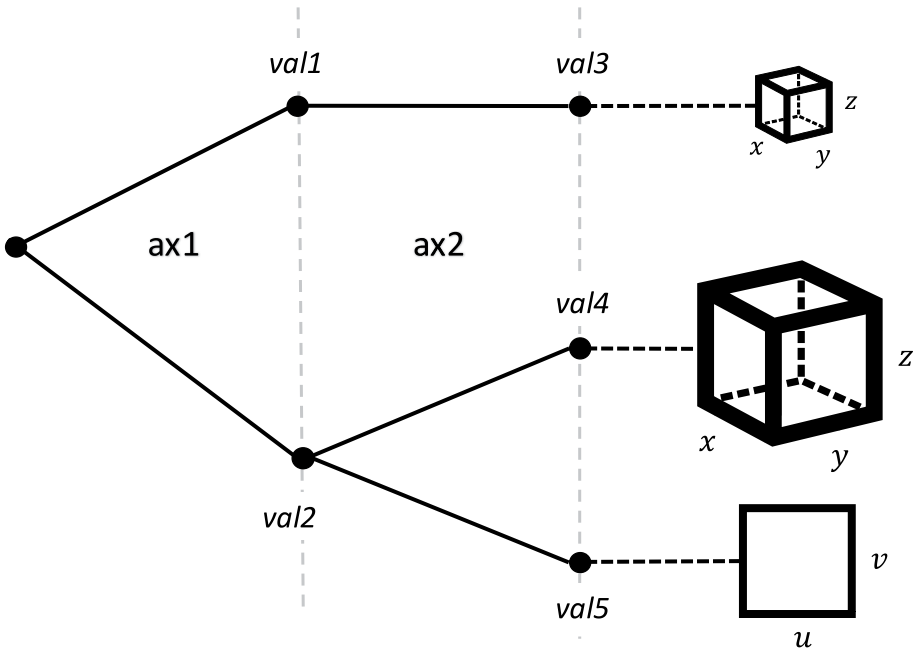


Fig. 2 Datacube represented as a non-regular imbalanced tree. Each tree layer represents different axis dimensions with the nodes being corresponding axis index values. Here, the first axis dimension is ax1, which has indices val1 and val2. The second axis dimension ax2 with indices val3, val4 and val5 then branches out onto datacubes of different sizes and dimensions, causing both imbalance and non-regularity in this tree

about the datacube structure. In particular, this suggests that there is a natural ordering of the axes, which we should follow when extracting data.

The imbalance in the datacube tree comes from the fact that some datacube axis can possibly have many more indices than others. In our example datacube above, imagine for instance that the u and v axis each only have 200 index values, whereas the x, y and z axis each have 1000 index values. This implies that the **val4** index has many more children than the **val5** index and makes this particular datacube very imbalanced.

Slicer

The core of the Polytope feature extraction algorithm is the *slicer*, which contains a novel slicing step on the datacube indices. The slicing algorithm introduced here is of particular relevance, as it supports non-orthogonal slicing across arbitrary ordered axes. This is in contrast to most current state of practice data extraction techniques, which often only cut boxes of data [34–36], whereas our slicing algorithm has the capability of cutting polytopes of data. Importantly, note that, as the algorithm introduced here is able to handle shapes of arbitrary dimensions, it can be used to extract both low- and high-dimensional data queries.

Concept

By leveraging results in the field of computational geometry, the slicing algorithm used in Polytope can extract any convex polytope from the original datacube. The underlying concept is that we successively slice the requested polytope along each axis in the datacube's natural axis ordering using hyperplanes, reducing the dimensionality of the polytope at each step until we are left with a list of all points contained in this polytope.

Ordered vs categorical axes

As mentioned earlier, the slicer handles ordered and categorical axes slightly differently.

In particular, categorical axes do not support range queries and thus we can only ask for specific values on these axes, instead of polytopes. For categorical axes, the algorithm therefore only has to check whether the queried indices exist in the datacube, as would happen in every other traditional extraction algorithm.

The true innovation of the Polytope extraction technique is its ability to handle arbitrary polytope requests, which it achieves by introducing a new slicing step along the ordered axes. Note however that this slicing technique only works on ordered axes for two reasons.

Firstly, since it is only possible to define and request ranges on ordered axes, it also only makes sense to define polytopes along such axes. Secondly, the slicing step introduced below only works on indices which can be interpolated. As we now explain, these are in fact precisely the ordered axes' indices. Indeed, note that, for the purposes of our algorithm, we assume that all of the ordered axes are measurable and linear axes, which can have continuous index values. We make this assumption even for ordered axes which are only truly countable with gaps between their indices. Because all ordered axes have some comparison operation, this is a valid assumption. This then implies that ordered axes support interpolation on their indices, which is needed for the slicing step described in the next subsection.

Slicing step

The actual slicing step is quite straightforward with the slicing mechanism merely consisting in finding the intersection of a polytope with a hyperplane along a datacube axis.

We first separate all vertices in the polytope into two separate sets, $P_{\leq}$ and $P_{\geq}$, each set consisting of points on either side of the hyperplane. For each pair of vertices $(p_{\leq}, p_{\geq})$, where $p_{\leq} \in P_{\leq}$ and $p_{\geq} \in P_{\geq}$, we take the intersection of the line passing through these two vertices and the hyperplane. This line-hyperplane intersection results in an intersection point which lies on the hyperplane. Once we have done this for all pairs, we therefore obtain a lower-dimensional polytope on the hyperplane, which is in fact just the intersection of the original polytope with the hyperplane, as wanted. This can be seen in Fig. 3 for a 2D example and in Fig. 4 for a 3D example.

As the original polytope is convex, this new intersection polytope is trivially also convex. As an optimisation step, we can thus take the convex hull of the intersection points at the end, using the QuickHull algorithm [37] for example. This does not change the lower-dimensional polytope because it is convex, but removes all interior vertices in its definition. As we slice high-dimensional polytopes, this can lead to major performance improvements. Indeed, without this last step, the number of vertex points in the polytope definition grows quadratically with each slice, which would considerably slow down the algorithm. However, even though this optimisation step can significantly help the performance of the algorithm, the line-hyperplane intersection operation is still quadratic in the number of polytope vertices. The algorithm's performance thus scales at least quadratically with the number of vertices, which will impact its runtime for input polytopes with a large number of vertices.

It is important to note here that this slicing step works on all ordered axes, without any specific constraints about the type of indices stored on these axes.

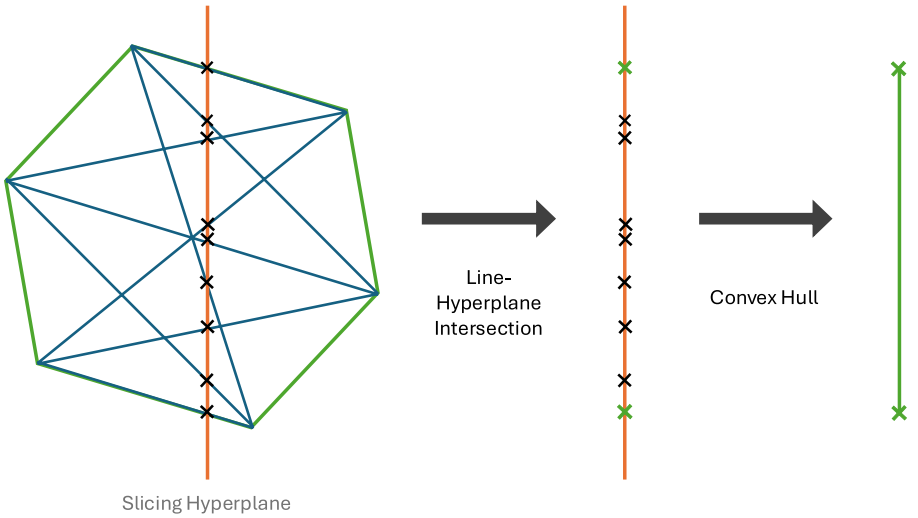


Fig. 3 2D example of the Polytope slicing mechanism on ordered axes

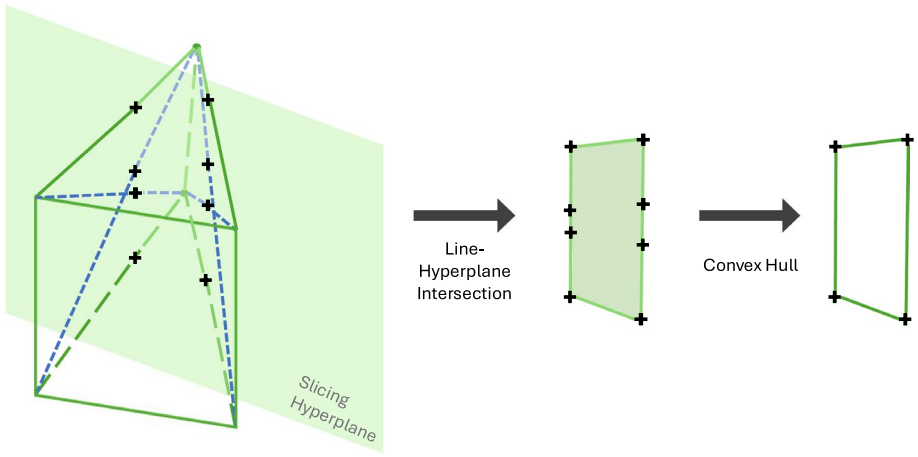


Fig. 4 3D example of the Polytope slicing mechanism on ordered axes

Index tree construction

To ensure that we slice through all the requested polytopes defined on different axes of the datacube, we need to carefully keep track of which step in the extraction we are in. The way we achieve this in the Polytope extraction technique is to iteratively build a trie compressed set of results, where the indices are the found datacube axis indices found inside of the polytope shape. In the following, we refer to this trie as an index tree.

We build the index tree by slicing along successive axes in the datacube's natural axis ordering one after another. For each axis of the datacube, we first find the polytopes defined on that axis. For each of these polytopes, we find the extents $(low, high)$ of the lowest and largest value of the polytope's boundary along the axis. We then collect all discrete indices contained within the extents $(low, high)$ on the axis and add them as children to the index tree. Next, we slice the necessary polytopes along each of the discrete datacube indices to obtain lower-dimensional polytopes. As shown in Fig. 5, these lower-dimensional polytopes are the intersection of the higher-dimensional polytopes with each of the axis indices slice hyperplanes. These new polytopes are the next polytopes we would like to now extract from the datacube. The algorithm therefore

continues as before on these lower-dimensional polytopes if they exist. This process is re-iterated in Algorithm 1.

Note that this works well on ordered axes. On categorical axes however, the slicing step is ill-defined as interpolation between indices is not possible. Nevertheless, recall that polytopes defined on categorical axes are in fact 1D points and thus instead of slicing, we only need to check whether those points exist in the datacube. Indeed, slicing does not matter in this case as the points are 1 dimensional and slicing, if it were well-defined, would therefore not produce any lower-dimensional polytopes anyway. We thus conclude that the process for constructing index trees presented in Algorithm 1 does in fact work well for categorical axes as well.

Algorithm 1 implies that we construct the index tree breadth-first (layer by layer), instead of depth-first (constructing branches one after the other). This approach ensures that the algorithm does not loose track of what values inside the requested polytopes have already been found. It thus ensures that users get back all the points that are contained in the shape they requested.

To gain a better understanding of how Algorithm 1 works in practice, we provide a detailed example of its mechanism on a small datacube in Appendix A.

Algorithm 1: Polytope Slicing Algorithm

Input: list of polytopes $\mathcal{P}$, **Output:** Index tree of results T

for polytope p in $\mathcal{P}$ **do**
    Remove duplicate points in p
end for
Instantiate root node r of an empty index tree T
Initialise a set $\mathcal{P}_{unsliced}$ of unsliced polytopes associated to r as $\mathcal{P}^r_{unsliced} := \mathcal{P}$
Initialise set of current index of tree nodes $current\_nodes := \{r\}$
for $axis$ in datacube axes **do**
    Initialise set of next nodes to consider $next\_nodes := \{\}$
    for node n in $current\_nodes$ **do**
        Find subset of polytopes $\mathcal{P}_{defined}$ on $axis$ from $\mathcal{P}^n_{unsliced}$
        for polytope p in $\mathcal{P}_{defined}$ **do**
            Find extents (low, $high$) of p on $axis$
            Find set $\mathcal{I}_{axis}$ of datacube indices between (low, $high$) on $axis$
            for index i in $\mathcal{I}_{axis}$ **do**
                Add a child c to n with c.axis := $axis$ and c.value := i
                Initialise the set $\mathcal{P}^c_{unsliced} := \mathcal{P}^n_{unsliced} \setminus \{p\}$
                if $axis$ is a categorical axis **then**
                    Skip
                else if $axis$ is an ordered axis **then**
                    Slice p along i to get lower-dimensional polytope $p_{reduced}$
                    Add $p_{reduced}$ to $\mathcal{P}^c_{unsliced}$
                end if
                Add c to $next\_nodes$
            end for
        end for
    end for
    Update $current\_nodes := next\_nodes$
end for
Return final index tree T

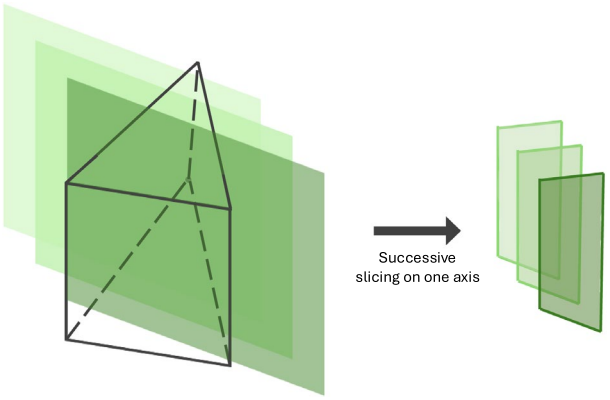


Fig. 5 Successive slicing of a polytope along one axis at different indices, resulting in a list of lower-dimensional polytopes

Applications

The Polytope data extraction algorithm has a wide range of interesting applications, from meteorology to healthcare. In this section, we first introduce the different Polytope interface levels before discussing some Polytope applications, describing specific examples and how Polytope has improved access to data in those cases.

Interface

To facilitate interaction with the Polytope feature extraction algorithm, which only accepts polytopes as input, different interfaces can be implemented. The Polytope interfaces allow the users to interact with the extraction algorithm. In particular, users will submit their request shapes and, after the algorithm has run, retrieve their desired data to and from these interfaces. To accommodate different types of users, several interface levels exist, which let users request a wide range of request shapes, from the low-level generic convex polytope to higher-level specialised requests. The two in-built low- and high-level Polytope interfaces are shown in Fig. 6. It is also possible to build domain-specific interfaces on top of these built-in interfaces, also shown in Fig. 6. Each level is built on top of another with the domain-specific interfaces using shapes from the high-level interface, which itself depends on the low-level interface.

These distinct interface levels are useful because depending on specific needs and familiarity with the Polytope extraction technique, users might want to request different types of shapes from our algorithm.

Through the domain-specific interfaces, users can request domain-specific functions. For example, a meteorological interface could be built to facilitate access to time-series, trajectories or country extraction, similar to the Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) [38] standard.

Through the built-in high-level interface, users can request primitive shapes, such as boxes or disks (approximated as dodecagons or 12-sided polygons), and then use constructive geometry operations, such as taking unions [39] or sweeping along a path [40], to build more complicated shapes. At this interface level, concave polygons are also supported by first triangulating the polygon and then taking the union of the resulting set of triangles. Finally, through the low-level interface, users can directly provide a list of convex n-dimensional polytopes, specified by a list of their vertices.

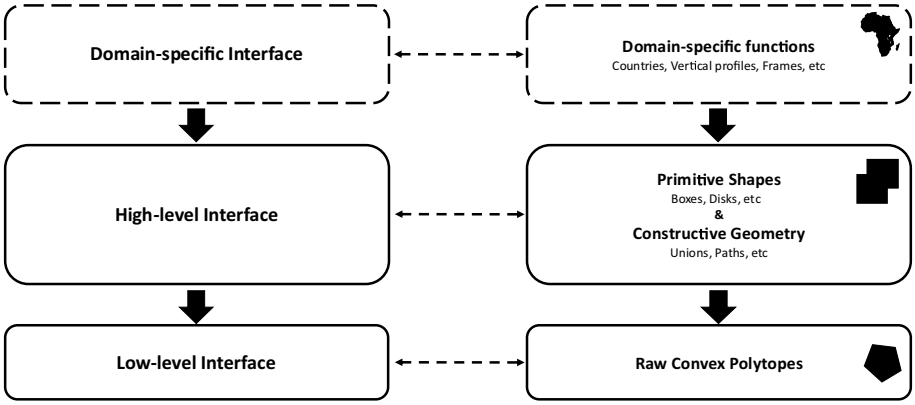


Fig. 6 Polytope interface levels

Each level is built on top of its lower-level counterpart, so that shapes in a higher level are always defined by shapes in one of the lower levels. This implies that shapes in any of the interface levels are in fact always defined as a combination of convex low-level polytopes. These low-level polytopes are the building blocks of all possible Polytope requests. The interface is responsible for decomposing all user request shapes into these base convex polytopes. In the slicing algorithm, we can then work only on these convex polytopes and take a unified approach towards slicing any user request shape.

Polytope use cases

We now describe some examples of how the Polytope feature extraction algorithm can be used in the fields of meteorology and healthcare.

Meteorology

At ECMWF, about 400 TiB of Numerical Weather Prediction (NWP) data are produced daily [41]. This data is usually represented as a datacube of 7 or 8 dimensions depending on the forecast type [42]. Over the next few years, following the pioneering work of the DestinE initiative [43] with planned resolution increases in the weather model, data production will grow to over a petabyte of data a day [44].

The current data extraction mechanism implemented at ECMWF in their Meteorological Archival and Retrieval System (MARS) [42, 45] is one of the traditional bounding box approaches. When a user wants to extract data on a country for example, they would need to send a request for a bounding box around that country. Moreover, the current extraction technique requires either full data fields or at very best bounding boxes of data fields to be read from the system even when users only request a smaller portion of data. With future petabyte-scale datacubes, this approach will become impractical, especially when trying to accommodate for thousands of users. The Polytope extraction technique helps alleviate many of the challenges faced by the system in this case. It makes returning data to users much more efficient because only the required bytes are read from the I/O system.

Below, we provide a few practical examples and use cases where Polytope might help meteorological data users extract data more efficiently.

Timeseries Imagine a user interested in extracting the temperature over Italy for the next two weeks. She would currently have to transpose the temperature fields along

the time axis to be able to then individually extract each temperature field at a given timestep. For each timestep, she would have to cut the shape of Italy from the bounding box she retrieved before finally getting the exact data she wanted. With the Polytope extraction technique, she can instead directly request the timeseries over Italy and get back only the precise bytes she is interested in. This is shown in Fig. 7, where the coloured points represent data points that are stored as byte indices within a data file and are accessed by the algorithm. Although Algorithm 1 only computes the coordinates of the data points, those coordinates can then be mapped to indices within a data file through a simple mapping. Polytope then accesses only those indices from the file instead of larger byte ranges, which are not of interest to the user. Note that compared to the 3D bounding box the user would currently retrieve, we see a data reduction of more than 73% when using Polytope. Furthermore, note that meteorological data users are usually more interested in extracting data over particular cities or specific points in space rather than whole regions, such as in [46, 47]. Since users currently first have to transpose their data and then retrieve bounding boxes around their locations of interest however, in most cases they directly extract data over broader regions than just the specific locations they would like to access. With the Polytope algorithm, expensive pre-processing manipulations before extraction, such as the ones discussed in [48], are not needed anymore and users only retrieve the relevant timeseries data from the datacube.

Flight Path Now, imagine a user interested in the flight conditions over his plane journey from Paris to New York. Using the current extraction technique, he would get back a 4 dimensional box over 3D space and time, containing much more data than what he is interested in. With the Polytope extraction technique, he will instead only get back the specific points he is interested in in the datacube without any need for post-processing, as shown in Fig. 8. Note that compared to the 4D box the user would currently get back, with Polytope, we experience a data reduction of more than 99.99%.

In both cases, the requested shapes are not axis-aligned and are therefore also not well approximated by bounding boxes. We thus see a significant data reduction when using the Polytope extraction technique compared to the traditional bounding box approach. Importantly, we observe that I/O is reduced when using the Polytope extraction

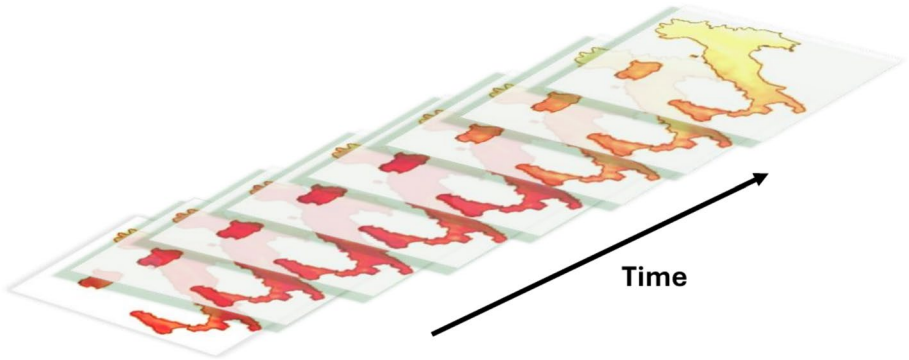


Fig. 7 Timeseries example, where the coloured points are all the points that were extracted using the Polytope algorithm

algorithm. Moreover users do not need to perform any post-processing on their data anymore in order to get their requested shape.

Healthcare

Similarly to the weather forecasting industry, the healthcare industry faces complex data handling challenges. Already in 2019, [49] estimated hospitals to generate tens of petabytes of data a year. As discussed in the previous example, working on this amount of data is extremely difficult and a tool like the Polytope algorithm could significantly help alleviate much of the difficulty involved.

A particular example of how Polytope can be used in the healthcare field is provided below.

> **Magnetic Resonance Imaging (MRI) Blood Vessel Detection** A clinically relevant application of MRI is the detection and characterization of plaque formation in (potential) stroke patients. This requires high-resolution scans using multiple MRI contrast weighting to comprehensively characterize the size and composition of plaque components. Using current extraction techniques, a clinician would have to download multiple entire MRI scan and then manually extract and compare the relevant data of interest from each of those scans. With the Polytope extraction technique however, it is possible to directly extract the required multi-contrast blood vessel data without further delay or expensive post-processing work, as is shown in Fig. 9 for a single high resolution black-blood vessel wall MRI dataset [50, 51].

Performance and scalability

Polytope has been designed to considerably decrease the computational cost of extracting non-orthogonal data from petabyte-sized hypercubes. In this section, we justify this claim by first analysing the performance of the Polytope algorithm and then investigating the data reductions achieved on practical use cases when using the Polytope algorithm instead of traditional extraction methods. The Polytope algorithm implemented here is written in Python and operates without parallel processing. All experiments are run on a macOS-12.2.1-arm64-arm-64bit operating system with a CPython v3.9.19 Python interpreter. We conclude this section by discussing these results and their significance.

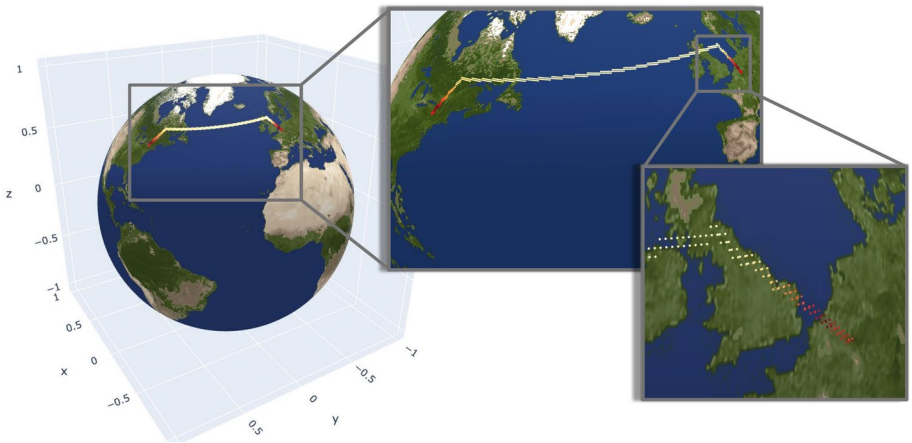


Fig. 8 Flight path example, where the coloured points are all the points that were extracted using the Polytope algorithm

Performance

There are many factors impacting the performance of the Polytope algorithm. To characterise it better, we identified some of the key features affecting how long the Polytope algorithm takes to extract points from datacubes: the number of extracted points, the dimension of the input shape and its geometry or how the input shape was constructed by the user. Note that, whilst there are many other important aspects which affect the algorithm's efficiency, such as the system's memory contingency, these are more system dependent and we thus do not study them here.

Consider two time quantities, the total slicing time and the algorithm run time. The slicing time is the total accumulated time spent just slicing, without constructing the index tree. The total algorithm run time however is the time it takes to perform all of Algorithm 1, including both the slicing time as well the time spent constructing the whole index tree. In Fig. 10, we have plotted both of these time quantities in different settings. In each subplot, we have varied one of the previously identified features and kept all others constant. This lets us gain an understanding of how each individual feature influences the performance of the Polytope algorithm. Note also that, although the algorithm influences data access patterns, we have not included the time taken in I/O to fetch the data from storage as this depends on the storage medium we use and is not strictly part of the Polytope algorithm.

In Fig. 10a, we first plot both the slicing time as well as the total algorithm run time for request shapes of different dimensions. We observe that the dimension of the shape does not significantly impact the algorithm's performance. This is due to the fact that, even when slicing higher-dimensional shapes, the Polytope algorithm spends most of its time slicing lower-dimensional polytopes. In fact, note that the number of polytopes to

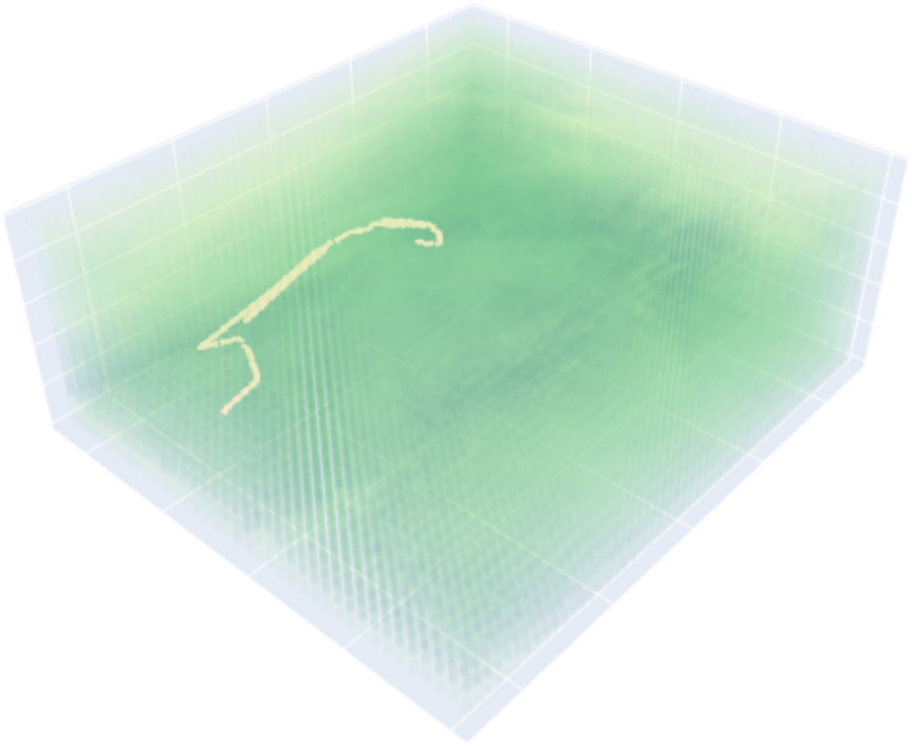


Fig. 9 MRI scan example, where the white line is an extracted blood vessel, reaching from the cavernous segment of the internal carotid artery to the end of the middle cerebral artery

process at each step in the algorithm quickly grows every time we slice to a lower dimension. For example, imagine we slice a 4D box which contains two indices on each dimension. We first have to perform 2 4D slices. This then gives us 2 3D boxes, which we now have to slice. Each of these 3D boxes still has two indices on each dimension along which we need to slice. For each 3D box, we thus have to perform 2 3D slices. Considering there are 2 3D boxes, this implies we have to perform 4 3D slices in total. If we continue this logic, at the end of the algorithm, we will have performed 2 4D slices, 4 3D slices, 8 2D slices and 16 1D slices. This illustrates why the lower-dimensional slices do in fact take up most of the algorithm's slicing time.

Furthermore, we note in Fig. 10a that the slicing time is much lower than the total algorithm run time. This is because Polytope is currently using the XArray [10] library for datacube implementation, and relies on XArray to look up discrete axis indices on the datacube. This is a step we believe still needs to be optimised by developing more efficient datacube look-up mechanisms and alternative datacube implementations. Meanwhile, in Figs. 10b-10d, we use the slicing time rather than the total algorithm run time to estimate Polytope's performance, thus excluding this dependency.

Figure 10b shows the behaviour of the slicing time in more detail. In particular, we notice that like the total algorithm run time in Fig. 10a, the slicing time does not depend on the dimension of the input shape. We also observe that the slicing time grows linearly with the number of datacube points the algorithm finds in the input shape. As discussed

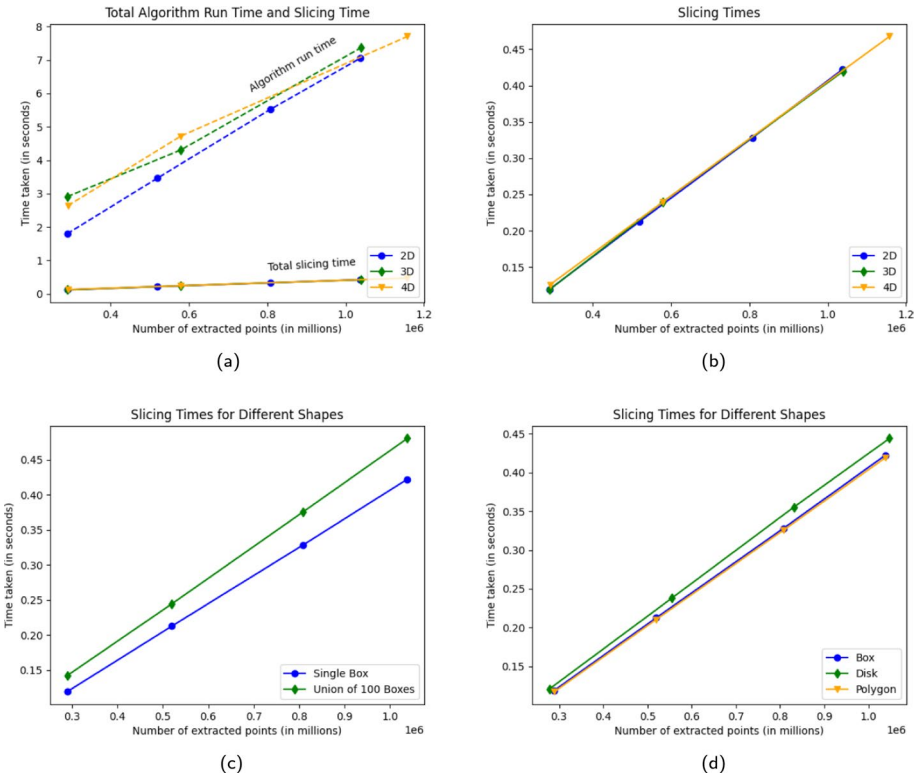


Fig. 10 Performance plots of the time taken to run the Polytope algorithm in function of the number of points extracted by the algorithm. In (**a**), we plot both the total algorithm run time (dashed lines) and the slicing time (solid lines) for different numbers of dimensions. In (**b**), we plot the slicing times for different numbers of dimensions. In (**c**) and (**d**), we plot the slicing times for different shape types. In (**c**), we focus on the effect of constructive geometry operations, whereas in (**d**), we focus on the various primitive shapes implemented in the high-level Polytope interface. Algorithm timings on an Apple M1 Pro chip (3.2 GHz, 8 cores) and 16GB DDR5

before, this is due to the fact that most of the slicing time is spent performing 1D slices. Indeed, increasing the number of points contained in the shape is effectively equivalent to increasing the number of 1D slices to perform to find those points. As it is those slices that make up most of the slicing time anyway, it is thus natural that the performance of the algorithm grows linearly with the number of points contained in the shape.

In Figs. 10c and 10d, we now investigate the impact of the input shape's geometry and how it was constructed by the user.

In Fig. 10c, we first study how constructing a shape by taking a union of smaller sub-shapes affects the performance of the algorithm compared to directly specifying the input shape as one single object in 2 dimensions. We see that the performance when the shape is constructed using unions is worse than when the shape is specified as one single object. This is due to the fact that when we request shapes as unions of sub-shapes, we first slice each sub-shape individually in the algorithm before combining the results of these steps into one single output. Because we first slice each sub-shape individually, we in fact slice along all the sub-shapes edges. As the sub-shapes touch along their edges, we thus end up slicing along the edges several times, which increases the slicing time compared to when this does not happen in the non-union shape case. This is relevant where the input geometry has been produced via triangulation or mesh generation.

In Fig. 10d, we finally analyse how Polytope's different 2 dimensional high-level interface primitive shapes: the box, disk and polygon shapes, influence the algorithm's performance. Here, we observe that the box and polygon shapes perform similarly while the disk shape has a slightly worse performance than the other two shapes. Because the polygon shape we inputted is actually a square, we can conclude from this observation that the algorithm's performance is mostly impacted by the number of vertices of the shape, as well as how "non-orthogonal" it is. In particular, if the shape has many edges cutting across some axes, then it is less likely that there will be duplicates when we compute the intersection points in the slicing step. We will thus then have to perform more slicing later on in the algorithm. The same logic holds if there are more vertices in the shape, which also increases the number of intersection points computed in the slicing step.

Bound on number of slices

As we just discussed, the time quantity of interest to us in evaluating the performance of the Polytope algorithm is the slicing time. The slicing time largely depends on a related quantity which is the number of slices performed during the algorithm. In this subsection, we quickly determine a theoretical upper bound to this related quantity.

Suppose we query an m dimensional shape using Polytope and let n_i be the maximal number of discrete indices stored along each of the $i = 1, \ldots, m$ axes of the datacube which are contained within the requested shape. Since we do not know a priori exactly how convex or non-box-like the requested shape is, we have to assume the worst-case scenario that the shape is in fact a box.

To find the datacube points within that worst-case box shape, the Polytope algorithm now first has to slice n_1 times along the first axis dimension. This produces n_1 $(m-1)$ -dimensional box shapes. We then have to slice each of these lower-dimensional box shapes n_2 times along the second dimension. This creates an additional $n_1 \times n_2$ slices.

If we continue this process up to the last 1D slices, we finally see that the number of slices performed during the Polytope algorithm, N_{slices} is bounded by

$$\begin{aligned} N_{slices} &\leq n_1 + n_1 \times n_2 + \cdots + n_1 \times \cdots \times n_m \\ &= \sum_{i=1}^{m} \prod_{j=1}^{i} n_j \,. \end{aligned}$$

We see that, as expected, this upper bound is dominated by the number $\prod_{i=1}^{m} n_i$ of 1D slices, which will take up most of the slicing time. As we saw in the previous subsection however, 1D slices are relatively inexpensive to perform and in all of the examples in Fig. 10, the slicing time remains under a second.

Data reductions

Although the Polytope algorithm represents an additional step to perform before extracting the data and it might thus at first glance seem like it has a much higher time complexity than traditional extraction approaches, it is important to remember the true purpose of the Polytope algorithm. Polytope is a tool which computes the precise bytes of data a user wants to access. Using this tool therefore implies that users only extract exactly the data points they need, which significantly reduces the number of points to be read from the I/O system compared to the alternative "bounding box" extraction techniques. This leads to faster data transfer times between systems, thus making this approach more time efficient when considered as a whole.

In this section, we analyse various data reduction statistics, which are indicators of the efficiency gains that can be achieved through Polytope on various datasets. The datasets used for the meteorological extractions are the NWP MARS archive datasets produced by ECMWF on a octahedral O1280 grid [42, 45]. For the extractions of the flight path from Paris to New York shown in Fig. 8, the point timeseries over a 14-day forecast range, and the vertical profile of a point extruded over 20 atmospheric pressure levels, the data is interpolated to a 0.25/0.25 regular latitude-longitude grid. For the box around Germany and country extractions, the data is interpolated to a higher-resolution 0.125/0.125 regular latitude-longitude grid instead. For the medical data extraction, the datasets shown are high-resolution black-blood vessel wall MRI images collected in [51]. The exact data reduction statistics achieved for the examples mentioned in the previous section in Figs. 7, 8 and 9 are shown in Table 1.

In Table 1, the first 3 columns show the number of bytes retrieved when using different extraction techniques. In particular, note the clear distinction between the first two columns which differentiate the bounding box approach described earlier from the state of practice extraction methods taken in the fields of meteorology and healthcare respectively. We refer to state of practice approaches here as full dataset file extractions, which are even less optimal than the bounding box approach. Indeed, it is important to note that one of the widely used extraction approaches in the field of meteorology for example is to extract whole fields, which are 2D arrays of latitude and longitude around the whole globe, from datacubes. Similarly, in the field of healthcare, MRI scans are currently stored as 3D images, which are completely extracted before any clipping operation is performed. The bounding box approach is thus already a clear improvement compared to these approaches. As we see in Table 1 however, Polytope performs even better

than the bounding box approach. This can be clearly observed in the fifth column, where we provide the reduction factor of the data retrieved when using the Polytope algorithm compared to the bounding box approach. The sixth column shows the total reduction of the data retrieved when using the Polytope algorithm compared to the state of practice extraction methods taken in the meteorology and healthcare fields. The final two columns then show the two slicing and total algorithm run times discussed above for each of our example shapes.

Along the rows, we differentiate between different types of shapes. On the first 3 rows, we first test the Polytope algorithm on shapes that are defined orthogonally along their axis and which could be directly extracted using the bounding box approach. For these 3 rows, as we see in the fifth column, using the Polytope algorithm instead of the bounding box approach does not reduce the size of the retrieved data further. Note however that running the Polytope algorithm in these three examples does not take significant time. In the latter 4 rows, we then experiment using the Polytope algorithm to retrieve more complicated non-orthogonal or axis-aligned shapes. Already for country shapes in 2D, we see that there is a significant data reduction when using the Polytope algorithm compared to the bounding box approach, with a reduction factor of up to 6 times in some cases. When considering higher dimensional shapes, and especially “path"-like shapes such as flight paths, we experience an even higher reduction factor. Indeed, in the 4D case of the flight path from Paris to New York mentioned above, about 350 times less data is returned to the users when using the Polytope algorithm instead of the bounding box approach. Again, note here that in most examples, the Polytope algorithm takes below a second to run whilst reducing the retrieved data size by a factor of at least 1000 compared to the state of practice approaches.

Importantly here, note that Polytope is able to perform the exact same orthogonal extractions as the bounding box approach in minimal time, whilst significantly outperforming the bounding box approach when extracting more complicated shapes. This suggests that the Polytope algorithm performs at least as well as the bounding box approach and thus makes it a strong competitor to this approach.

Discussion

Note that in subsection 5.1 above, in Fig. 10a, we did not include the time spent extracting data from the datacube. This is because this data extraction time is very dependent on the storage medium on which the algorithm is run. We expect that the cost of performing the Polytope algorithm will be significantly less than the savings made by retrieving less data. In particular, we suspect that hardware that supports high performance random-read, such as flash based devices for example, will benefit massively from the Polytope algorithm. To take advantage of sequential data access, where possible, calls to storage are batched together into byte ranges so that the access pattern aligns better with the data storage layout. This optimisation step allows the Polytope algorithm to perform well for both single point extraction, as well as area extractions, without incurring significant additional costs compared to block extractions.

Compared to state of practice methods, the Polytope algorithm is an additional step in the extraction process which takes time to run. However, as we saw in this section, and in Fig. 10 especially, the Polytope algorithm is efficient and scalable, being able to locate more than a million points in less than half a second. Moreover, as already mentioned,

Table 1 Polytope data reduction and performance

Example Shape	Data retrieved with state of practice approach	Data retrieved with bounding box approach	Data retrieved with Polytope algorithm	Reduction factor compared to state of practice approach	Reduction factor compared to bounding box approach	Slicing time	Total algorithm run time
Box around germany	50.4 MB	44 KB	44 KB	1173 ×	1 ×	2.3e-3 s	0.03 s
Timeseries of London over 14 days	5.5 GB	896 B	896 B	6591049 ×	1 ×	1.4e-4 s	0.13 s
Vertical profile over 20 layers - Rome	1 GB	800 B	800 B	1342177 ×	1 ×	4.6e-5 s	0.02 s
Country shape of france	50.4 MB	67.7 KB	32.3 KB	1598 ×	2 ×	0.03 s	0.94 s
Country shape of norway	50.4 MB	171.4 KB	29.9 KB	1726 ×	6 ×	0.06 s	1.97 s
Flight path from Paris to New York	7.9 GB	247.3 MB	4.9 KB	1690561 ×	51681 ×	0.07 s	0.18 s
MRI blood vessel	1 GB	1.5 MB	4.5 KB	233017 ×	341 ×	0.10 s	0.35 s

Algorithm timings on an Apple M1 Pro chip (3.2 GHz, 8 cores) and 16GB DDR5. Note that for completeness, both slicing and total algorithm run time are included, although the slicing time is more indicative of the Polytope algorithm performance, as discussed in subsection 5.1

when discussing performance of the algorithm, it is especially important to also consider its wider role in the total extraction process. As we saw in Table 1, the Polytope algorithm allows users to extract much less data than they would have done using a more traditional approach. As reading and returning data is usually a costly operation, it implies that, when incorporated in a complete extraction pipeline, the Polytope algorithm will make data extraction more efficient than the current state of practice. The slightly more expensive slicing mechanism inside the Polytope algorithm will thus be outweighted by the actual performance improvement of the whole data extraction pipeline.

To quantify the true performance and benefits of the Polytope algorithm, we have performed a more in-depth analysis of its behaviour within a complete data extraction framework in [52]. In this work, we investigated the performance and scalability of the Polytope algorithm on the FDB meteorological data stores [53]. We found that for domain-specific feature shapes of interest, such as point timeseries, an end-to-end system using the Polytope algorithm extracts data about 1 to 2 orders of magnitude faster than traditional data extraction methods. In particular, the analysis shows that point timeseries across the whole forecast length are extracted in a fraction of a second, whereas extracting the equivalent full atmospheric fields from the FDB takes a few seconds. When scaled to higher dimensions, this gap only increases with a Polytope extraction for a point timeseries over all forecast ensemble members taking around 4 seconds, whilst accessing the corresponding full atmospheric fields from the FDB takes nearly 2 minutes. On top of faster extraction times, we also observe in the paper that Polytope reduces byte access to a small fraction of the full atmospheric fields stored on the FDB. For a point timeseries extraction for example, Polytope only reads a few hundred bytes, whereas the equivalent full fields accumulate to around a gigabyte of data. Compared to full FDB field extractions, we therefore conclude that using Polytope will likely lead to a significantly reduced resource usage of the overall system. The exact savings will depend on the access patterns that are requested by the users and on the system hardware specifications. A more detailed analysis is beyond the scope of this publication.

Conclusion

In this paper, we introduced a new data extraction algorithm called Polytope, which has the capability of extracting arbitrary geometrical shapes from a datacube. This new technique allows users to directly compute the precise bytes of interest to them before requesting these bytes from a datacube. This approach leads to many benefits, both for the users and the data providers. For data providers, much less I/O is needed whereas for users, the need for further post-processing after extraction is alleviated. We described the structure of this novel extraction algorithm and explained in more detail some of its key features. We then showed a few use cases of the Polytope extraction technique before finally analysing the performance of this method. Future steps include performing a more in-depth analysis of the algorithm performance, as well as a rigorous discussion of Polytope's use cases in different scientific fields.

Appendix A: Polytope algorithm example

Take a pyramid with vertices [[0, 0, 0], [0, 1, 0], [1, 1, 0], [1, 0, 0], [0.5, 0.5, 1]]. The datacube we slice against is a regular 3-dimensional point cloud with points at the intersections of the lines $x = (0, 0.25, 0.5, 0.75, 1)$, $y = (0, 0.25, 0.5, 0.75, 1)$ and $z = (0, 0.25, 0.5, 0.75, 1)$.

The first axis we slice along is the *z*-axis. The extents of the pyramid along the *z*-axis is (0, 1). Between $z = 0$ and $z = 1$, we have datacube points at $z = (0, 0.25, 0.5, 0.75, 1)$. Our algorithm thus creates index nodes $(z = 0), (z = 0.25), (z = 0.5), (z = 0.75)$ and $(z = 1)$ in the index tree and then slices the pyramid along each of those *z* values.

The first resulting 2-dimensional square of the intersection between the pyramid and the plane $z = 0$ is the square in $x - y$ space with vertices [[0, 0], [0, 1], [1, 1], [1, 0]]. We store this reduced polytope in the corresponding tree node $(z = 0)$. We then slice along the *y*-axis. The extents of the reduced 2-dimensional square along the *y*-axis is (0, 1). Between $y = 0$ and $y = 1$, we have datacube points at $y = (0, 0.25, 0.5, 0.75, 1)$. Our algorithm thus creates sub-index nodes $(y = 0), (y = 0.25), (y = 0.5), (y = 0.75)$ and $(y = 1)$ as children of the node $(z = 0)$ and then slices the square along each of those *y* values.

The first resulting line segment of the intersection between the square and the line $y = 0$ is the line segment in *x* space with vertices [0, 1]. We store this reduced polytope in the corresponding tree node $(z = 0, y = 0)$. The final and last axis to slice is the *x*-axis here. The extents of the line segment along the *x*-axis is (0, 1). Between $x = 0$ and $x = 1$, we have datacube points at $x = (0, 0.25, 0.5, 0.75, 1)$. We thus create the leaf nodes of this sub-branch $(x = 0), (x = 0.25), (x = 0.5), (x = 0.75)$ and $(x = 1)$.

We proceed similarly for the other *y* sub-index nodes $(y = 0.25), (y = 0.5), (y = 0.75)$ and $(y = 1)$, where we also have leaf nodes $(x = 0), (x = 0.25), (x = 0.5), (x = 0.75)$ and $(x = 1)$ for each of those.

This finally gives the complete first branch of the result tree as shown in Fig. 11.

The second resulting 2-dimensional square of the intersection between the pyramid and the plane $z = 0.25$ is the square in $x - y$ space with vertices [[0.125, 0.125], [0.125, 0 .875], [0.875, 0.875], [0.875, 0.125]]. We store this reduced polytope in the corresponding tree node $(z = 0.25)$. We then slice along the *y*-axis. The extents of the reduced 2-dimensional square along the *y*-axis is (0.125, 0.875). Between $y = 0.125$ and $y = 0.875$, we have datacube points at $y = (0.25, 0.5, 0.75)$. Our algorithm thus creates sub-index nodes $(y = 0.25), (y = 0.5)$ and $(y = 0.75)$ as children of the node $(z = 0.25)$ and then slices the square along each of those *y* values. Slicing the square along each of these *y* values, we create the leaf nodes of each of those sub-branches $(x = 0.25), (x = 0.5)$ and $(x = 0.75)$. We then get the second branch of the result tree as shown in Fig. 12.

The third resulting 2-dimensional square of the intersection between the pyramid and the plane $z = 0.5$ is the square in $x - y$ space with vertices [[0.25, 0.25], [0.25, 0.75], [0. 75, 0.75], [0.75, 0.25]]. We store this reduced polytope in the corresponding tree node $(z = 0.5)$. We then slice along the *y*-axis. The extents of the reduced 2-dimensional square along the *y*-axis is (0.25, 0.75). Between $y = 0.25$ and $y = 0.75$, we have datacube points at $y = (0.25, 0.5, 0.75)$. Our algorithm thus creates sub-index nodes $(y = 0.25), (y = 0.5)$ and $(y = 0.75)$ as children of the node $(z = 0.5)$ and then slices the square along each of those *y* values. Slicing the square along each of these *y* values, we create the leaf nodes of each of those sub-branches $(x = 0.25), (x = 0.5)$ and $(x = 0.75)$. We then get the third branch of the result tree as shown in Fig. 13.

The fourth resulting 2-dimensional square of the intersection between the pyramid and the plane $z = 0.75$ is the square in $x - y$ space with vertices [[0.375, 0.375], [0.375, 0.62 5], [0.625, 0.625], [0.625, 0.375]]. We store this reduced polytope in the corresponding tree node $(z = 0.75)$. We then slice along the *y*-axis. The extents of the reduced 2-dimen-

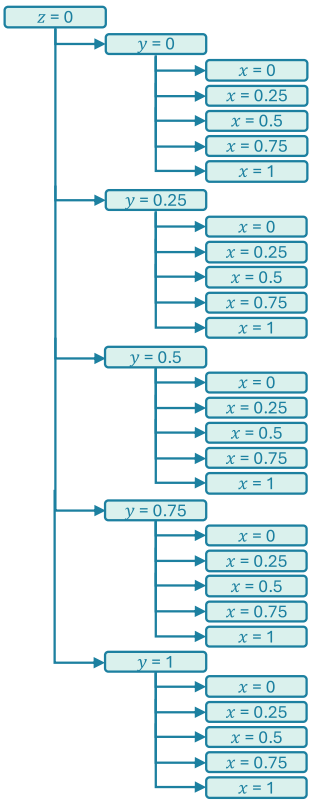


Fig. 11 First tree branch

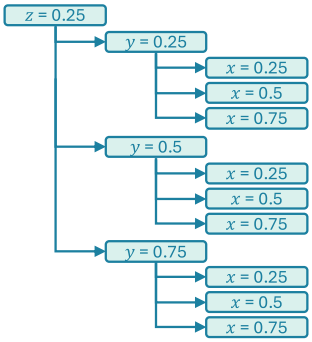


Fig. 12 Second tree branch

sional square along the y-axis is (0.375, 0.625). Between $y = 0.375$ and $y = 0.625$, we have datacube points at $y = (0.5)$. Our algorithm thus creates the sub-index node $(y = 0.5)$ as a child of the node $(z = 0.75)$ and then slices the square along this y value. Slicing the square along this y value, we create the leaf node of this sub-branch $(x = 0.5)$. We then get the fourth branch of the result tree as shown in Fig. 14.

Finally, the fifth resulting 2-dimensional square of the intersection between the pyramid and the plane $z = 1$ is the point in $x - y$ space [0.5, 0.5]. We store this reduced polytope in the corresponding tree node $(z = 1)$. We then slice along the y-axis. The extents of the point along the y-axis is (0.5, 0.5). Between $y = 0.5$ and $y = 0.5$, we have datacube points at $y = (0.5)$. Our algorithm thus creates the sub-index node $(y = 0.5)$ as a child of the node $(z = 1)$ and then slices the square along this y value. Slicing the square along

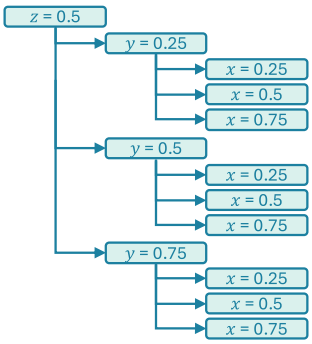


Fig. 13 Third tree branch

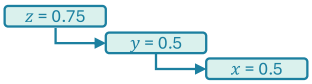


Fig. 14 Fourth tree branch

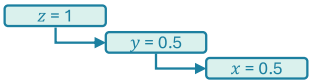


Fig. 15 Fifth tree branch

this y value, we create the leaf node of this sub-branch $(x = 0.5)$. We then get the fourth and final branch of the result tree as shown in Fig. 15.

At the end of the algorithm, we have now found all datacube points contained in the original pyramid, which are stored in our final result tree, as expected.

Abbreviations

ECMWF	European centre for medium range weather forecasts
ESEE	ECMWF software enginE
DestinE	Destination earth
DTE	Digital twin engine
OGC	Open geospatial consortium
EDR	Environmental data retrieval
NWP	Numerical weather prediction
MARS	Meteorological archival and retrieval system
MRI	Magnetic resonance imaging

Acknowledgements
The authors would like to thank Matthijs de Buck and the Nuffield Department of Clinical Neurosciences at the University of Oxford for providing data for the MRI Blood Vessel Detection use case. The authors are also very grateful to ECMWF, the European Union's Destination Earth initiative and multiple European Commission projects and programmes which have helped fund this work.

Author contributions
ML, JH, SS, ED and TQ contributed to the initial design and concept of the algorithm. ML and JH contributed to the implementation of the algorithm. ML conducted the performance and scalability analysis. ML wrote the main manuscript text. JH, TQ and MS reviewed the manuscript.

Funding
The work presented in this paper has been co-funded in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Competing interests
The authors declare no competing interests.

Received: 21 December 2024 / Accepted: 8 October 2025
Published online: 01 November 2025

References

1. Yaqoob I, Hashem IAT, Gani A, Mokhtar S, Ahmed E, Anuar NB, et al. Big data: from beginning to future. Int J Inf Manage. 2016;36(6):1231–47.
2. Bauer P, Quintino T, Wedi N, Bonanni A, Chrust M, Deconinck W, Diamantakis M, Düben P, English S, Flemming J et al. The ECMWF Scalability Programme: Progress and Plans. ECMWF 2020. https://www.ecmwf.int/node/19380
3. Data: a small four-letter word which has grown exponentially to such a big value. https://www.deloitte.com/cy/en/Industries/technology/perspectives/data-grown-big-value.html. Accessed on 3 Oct 2024
4. Baumann P, Misev D, Merticariu V, Huu BP. Array databases: concepts, standards, implementations. J Big Data. 2021;8:1–61.
5. Baumann P, Misev D, Merticariu V, Huu, BP. Datacubes: Towards space/time analysis-ready data. Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management, 2019;269–299
6. Killough B. Overview of the open data cube initiative. In: IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, 2018;pp. 8629–8632 . IEEE
7. Baumann P. Datacube standards and their contribution to analysis-ready data. In: IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, 2018;pp. 2051–2053 . IEEE
8. Wang Z, Chu Y, Tan K-L, Agrawal D, Abbadi AE, Xu X. Scalable data cube analysis over big data. arXiv preprint arXiv:1311.5663 2013.
9. Higham DJ, Higham NJ. MATLAB Guide. vol. 150. SIAM 2016.
10. Hoyer S, Hamman J. Xarray: ND labeled arrays and datasets in Python. J Open Res Softw. 2017;5(1):10.
11. Xtensor Stack: Xtensor Documentation. In *Xtensor* (Version 0.24.6). Retrieved from Read the Docs: https://xtensor.readthedocs.io/en/latest/. Accessed on 10 Feb 2023
12. Melton J, Simon AR. SQL: 1999: understanding relational language components. Elsevier 2001.
13. Snodgrass R. The temporal query language TQuel. ACM Trans Database Syst. 1987;12(2):247–98.
14. Obe RO, Hsu LS. PostgreSQL: up and running: a practical guide to the advanced open source database. O'Reilly Media: Inc; 2017.
15. Baumann P, Furtado P, Ritsch R, Widmann N. The RasDaMan approach to multidimensional database management. In: Proceedings of the 1997 ACM Symposium on Applied Computing, 1997;pp. 166–173
16. Fiore S, D'Anca A, Elia D, Palazzo C, Williams D, Foster I, Aloisio G. Ophidia: a full software stack for scientific data analytics. In: 2014 International Conference on High Performance Computing & Simulation (HPCS), 2014;pp. 343–350 . IEEE
17. Gosink L, Shalf J, Stockinger K, Wu K, Bethel W. HDF5-FastQuery: Accelerating complex queries on HDF datasets using fast bitmap indices. In: 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), 2006;pp. 149–158 . IEEE
18. Taylor KE. Ezget: A library of fortran subroutines to facilitate data retrieval. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States) 1996.
19. Gray J, Chaudhuri S, Bosworth A, Layman A, Reichart D, Venkatrao M, et al. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data Min Knowl Discov. 1997;1:29–53.
20. Frihida A, Marceau DJ, Thériault M. Extracting and visualizing individual space-time paths: an integration of GIS and KDD in transport demand modeling. Cartogr Geogr Inf Sci. 2004;31(1):19–28.
21. Baumann P. Management of multidimensional discrete data. VLDB J. 1994;3(4):401–44.
22. Leuridan M, Warde A, Hawkes J, Varndell J, Tsrunchev P, Figala D. ecmwf/polytope: 1.0.21. https://doi.org/10.5281/zenodo.14537049 .
23. Geenen T, Wedi N, Milinski S, Hadade I, Reuter B, Smart S, et al. Digital twins, the journey of an operational weather system into the heart of Destination Earth. Procedia Computer Science. 2024;240:99–108.
24. Wolfe P. Finding the nearest point in a polytope. Math Program. 1976;11:128–49.
25. Thompson AC. Convex sets. In: Meyers, R.A. (ed.) Encyclopedia of Physical Science and Technology (Third Edition), Third edition edn., 2003;pp. 717–737. Academic Press, New York. https://doi.org/10.1016/B0-12-227410-5/00146-0. https://www.sciencedirect.com/science/article/pii/B0122274105001460
26. Chazelle B, Palios L. Decomposition algorithms in geometry. In: Algebraic Geometry and Its Applications: Collections of Papers from Shreeram S. Abhyankar's 60th Birthday Conference, 1994;pp. 419–447 . Springer
27. Owen SJ. A survey of unstructured mesh generation technology. IMR. 1998;239:267.
28. Guttman A. R-trees: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, 1984;pp. 47–57
29. Obe R, Hsu L. PostGIS in action. Simon and Schuster 2021.
30. Stolte C, Tang D, Hanrahan P. Multiscale visualization using data cubes. IEEE Trans Vis Comput Graph. 2003;9(2):176–87.
31. Purss MB, Peterson PR, Strobl P, Dow C, Sabeur ZA, Gibb RG, et al. Datacubes: a discrete global grid systems perspective. Cartographica. 2019;54(1):63–71.
32. Harzheim E. Ordered sets. vol. 7. Springer Science & Business Media 2005.
33. Otoo EJ, Wang H, Nimako G. Multidimensional Sparse Array Storage for Data Analytics. In: 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016;pp. 1520–1529 . IEEE
34. Makris A, Tserpes K, Spiliopoulos G, Anagnostopoulos D. Performance Evaluation of MongoDB and PostgreSQL for Spatio-temporal Data. In: EDBT/ICDT Workshops 2019.

35. Su Y, Agrawal G. Supporting user-defined subsetting and aggregation over parallel NetCDF datasets. In: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012;pp. 212–219 . IEEE
36. Su Y, Agrawal G, Woodring J. Indexing and parallel query processing support for visualizing climate datasets. In: 2012 41st International Conference on Parallel Processing, 2012;pp. 249–258 . IEEE
37. Barber CB, Dobkin DP, Huhdanpaa H. The quickhull algorithm for convex hulls. ACM Trans Math Softw. 1996;22(4):469–83.
38. Burgoyne M, Blodgett D, Heazel C, Little C. OGC API-Environmental Data Retrieval Standard. Open Geospatial Consortium Inc., Wayland, MA, USA, OpenGIS® Implementation Specification OGC
39. Ricci A. A constructive geometry for computer graphics. Comput J. 1973;16(2):157–60.
40. Martin RR, Stephenson P. Sweeping of three-dimensional objects. Computer-Aided Design. 1990;22(4):223–34.
41. ECMWF: Key facts and figures. https://www.ecmwf.int/en/about/media-centre/key-facts-and-figures. Accessed on 29 Sep 2024
42. ECMWF: MARS Catalogue. https://apps.ecmwf.int/mars-catalogue/. Accessed on 29 Sep 2024
43. Wedi N, Quintino T, Modigliani U, Baousis V, Geenen T, Sandu I, et al. Destination Earth: Digital Twins of the Earth System. Copernicus Meetings: Technical report; 2022.
44. Wedi N, Bauer P, Sandu I, Hoffmann J, Sheridan S, Cereceda R, et al. Destination Earth: High-Performance Computing for Weather and Climate. Computing in Science & Engineering. 2022;24(6):29–37.
45. Raoult B. Architecture of the new MARS server. https://www.ecmwf.int/sites/default/files/elibrary/1997/11839-architecture-new-mars-server.pdf. Accessed on 11 Feb 2023
46. Guo Y, Punnasiri K, Tong S. Effects of temperature on mortality in Chiang Mai city, Thailand: a time series study. Environ Health. 2012;11:1–9.
47. Chen R, Yin P, Wang L, Liu C, Niu Y, Wang W, Jiang Y, Liu Y, Liu J, Qi J et al. Association between ambient temperature and mortality risk and burden: time series study in 272 main Chinese cities. BMJ 2018;**363**
48. Ordonez C. Data set preprocessing and transformation in a database system. Intell Data Anal. 2011;15(4):613–31.
49. 4 ways data is improving healthcare. https://www.weforum.org/agenda/2019/12/four-ways-data-is-improving-healthcare. Accessed on 24 Apr 2023
50. Viessmann O, Li L, Benjamin P, Jezzard P. T2-weighted intracranial vessel wall imaging at 7 tesla using a DANTE-prepared variable flip angle turbo spin echo readout (DANTE-SPACE). Magn Reson Med. 2017;77(2):655–63.
51. Buck MH, Hess AT, Jezzard P. Simulation-based optimization and experimental comparison of intracranial T2-weighted DANTE-SPACE vessel wall imaging at 3T and 7T. Magn Reson Med. 2024;92(5):2112–26.
52. Leuridan M, Bradley C, Hawkes J, Quintino T, Schultz M. Performance analysis of an efficient algorithm for feature extraction from large scale meteorological data stores. In: Proceedings of the Platform for Advanced Scientific Computing Conference, 2025;pp. 1–9
53. Smart SD, Quintino T, Raoult B. A high-performance distributed object-store for exascale numerical weather prediction and climate. In: Proceedings of the Platform for Advanced Scientific Computing Conference, 2019;pp. 1–11

Publisher's Note

3 Serving Features on Meteorological Data Stores

In the previous chapter, we introduced a feature extraction algorithm built on top of xarray DataArray objects. While this is effective in practice, the approach does not reach its full potential, as the entire dataset must still be loaded into memory before extraction. As a result, it offers limited I/O benefits and remains conceptually similar to post-processing clipping methods, although in a more general form. To achieve substantial I/O reductions in operational weather systems, the Polytope algorithm must be integrated directly into existing meteorological data infrastructures and NWP data stores. In the following paper [69], we adapt and deploy the Polytope algorithm on top of ECMWF's FDB data stores [105, 106]. We present a unified data extraction framework that combines the Polytope feature extraction algorithm with direct byte-level access to the underlying data stores. This byte-level access is enabled by the GribJump software, which allows the system to jump directly to the required bytes within the FDB. Together, these components remove the need to read and load entire dataset files when serving individual user requests. We then provide a detailed analysis of the system and its individual components, evaluating performance and scalability on real-world NWP systems at ECMWF as part of the Destination Earth initiative. Finally, we demonstrate the framework's effectiveness through practical use cases, such as the extraction of regional time series.

This work was published in the 2025 proceedings of the Platform for Advanced Scientific Computing (PASC) and the source code for the Polytope [75] and GribJump [32] software are both publicly available on GitHub. I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *C. Bradley*: Conceptualisation, Software; *J. Hawkes*: Conceptualisation, Writing-review & editing; *M. Schultz*: Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

Performance Analysis of an Efficient Algorithm for Feature Extraction from Large Scale Meteorological Data Stores

Mathilde Leuridan
European Center for Medium-Range Weather Forecasts (ECMWF)
and University of Cologne
mathilde.leuridan@ecmwf.int

Christopher Bradley
European Center for Medium-Range Weather Forecasts (ECMWF)
chris.bradley@ecmwf.int

James Hawkes
European Center for Medium-Range Weather Forecasts (ECMWF)
james.hawkes@ecmwf.int

Tiago Quintino
European Center for Medium-Range Weather Forecasts (ECMWF)
tiago.quintino@ecmwf.int

Martin Schultz
Forschungszentrum Juelich (FZJ)
and University of Cologne
m.schultz@fz-juelich.de

ABSTRACT

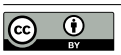
In recent years, Numerical Weather Prediction (NWP) has undergone a major shift with the rapid move towards kilometer-scale global weather forecasts and the emergence of AI-based forecasting models. Together, these trends will contribute to a significant increase in the daily data volume generated by NWP models. Ensuring efficient and timely access to this growing data requires innovative data extraction techniques. As an alternative to traditional data extraction algorithms, the European Centre for Medium-Range Weather Forecasts (ECMWF) has introduced the Polytope feature extraction algorithm. This algorithm is designed to reduce data transfer between systems to a bare minimum by allowing the extraction of non-orthogonal shapes of data. In this paper, we evaluate Polytope's suitability as a replacement for current extraction mechanisms in operational weather forecasting. We first adapt the Polytope algorithm to operate on ECMWF's FDB (Fields DataBase) meteorological data stores, before evaluating this integrated system's performance and scalability on real-time operational data. Our analysis shows that the low overhead of running the Polytope algorithm, which is in the order of a few seconds at most, is far outweighed by the benefits of significantly reducing the size of the extracted data by up to several orders of magnitude compared to traditional bounding box methods. Our ensuing discussion focuses on quantifying the strengths and limitations of each individual part of the system to identify potential bottlenecks and areas for future improvement.

CCS CONCEPTS

• **Mathematics of computing** → **Mathematical software performance**; • **Information systems** → **Data access methods**; **Extraction, transformation and loading**.

KEYWORDS

Data Extraction, Data Management, Numerical Weather Prediction



PASC '25, June 16–18, 2025, Brugg-Windisch, Switzerland

ACM ISBN 979-8-4007-1886-1/2025/06
https://doi.org/10.1145/3732775.3733573

ACM Reference Format:
Mathilde Leuridan, Christopher Bradley, James Hawkes, Tiago Quintino, and Martin Schultz. 2025. Performance Analysis of an Efficient Algorithm for Feature Extraction from Large Scale Meteorological Data Stores. In *Platform for Advanced Scientific Computing Conference (PASC '25), June 16–18, 2025, Brugg-Windisch, Switzerland*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3732775.3733573

1 INTRODUCTION

Meteorological data ranks among one of the largest sources of scientific data today. Modern weather forecasting models now generate high-resolution data for hundreds of parameters and ensemble members, extending over medium-range periods of up to 10-15 days. Each day, these models produce hundreds of terabytes, forming massive high-dimensional datacubes. As we push towards kilometer-scale global weather forecasting [19, 27] and increasingly leverage AI [11, 12] to enable more frequent model runs, the volume of this data is set to continue growing exponentially over the coming years.

Once available, this data is used in a wide range of applications, from disaster prevention to climate research or resource management. In many cases, users want to extract data over specific regions of interest. For instance, one user might want to request precipitation levels for a local area to evaluate the risk of flash floods [1], while another needs wind speed data over wind farm locations in the Baltic Sea to estimate energy production [20] over the next two days.

In most current systems however, data access is limited to orthogonal queries [6, 8]. Users are required to specify ranges along the datacube's metadata axes, rather than being able to extract non-orthogonal shapes, such as the ones described above, directly. As a result, users often retrieve bounding boxes around their area of interest, leading to the extraction of excess data.

To address this limitation, ECMWF introduced a feature extraction algorithm called Polytope [13]. Polytope computes the exact metadata indices along the datacube's axes that fall within an arbitrary user-defined shape. When combined with a data store that enables individual byte access, this algorithm allows the system to read and retrieve only the specific bytes of interest to the user. Polytope is an integral component of both the ECMWF Software EnginE (ESEE), which is the smart software layer enabling all data provision and

workflow management services at ECMWF, and the Destination Earth Digital Twin Engine (DTE) [7].

In this paper, we integrate the Polytope feature extraction algorithm[1] with the FDB meteorological data store [21, 22]. In particular, we introduce a new component, GribJump[2], which enables single byte access to data stored in GRIB format [25] within the FDB. This operation, which was not previously supported by the FDB, is an essential step towards unlocking Polytope's full functionality of accessing only data within a given user-specified shape. We then perform a detailed study of the performance and scalability of this integrated system. There are many different aspects to consider during this analysis, from algorithmic to more technical considerations. We identify these factors and investigate how they affect the overall system's performance. Thereafter, we explore the impact of this data extraction technique on broader operational NWP systems, particularly in terms of reducing the data transfer between such systems and their users. We finally discuss the implications of our results before highlighting possible future improvements to Polytope and GribJump.

2 FEATURE EXTRACTION ON METEOROLOGICAL DATA STORES

2.1 Meteorological Data Storage

NWP models account for the daily production of hundreds of terabytes of data. ECMWF's Integrated Forecasting System (IFS) [16] for example, generates around 400 terabytes (TB) of meteorological data every day [5]. With expected improvements in model resolution, such as those planned under the European Union's Destination Earth initiative [10, 24], daily data output is expected to exceed a petabyte in the coming years.

Once generated, this data is stored in a meteorological object store for several days to remain easily accessible to users. A widely used system for this purpose is the FDB [21, 22], which supports ECMWF operations as well as large European projects such as Destination Earth. As an indexed data store, the FDB is optimized for efficient data retrieval using pre-defined metadata schemas. All data within the FDB is stored in GRIB (General Regularly distributed Information in Binary form) [25] messages, where each message represents a distinct atmospheric field. The data within a message is fully specified by its associated metadata, such as the date and time at which the data was generated or the vertical level and time step of the forecast to which this data corresponds. Additionally, every message contains a latitude-longitude grid, onto whose indices data values are mapped. This allows the data to be represented spatially across the Earth.

Many operational models are computed on high-resolution grids, containing millions of points. The O1280 octahedral grid used in ECMWF's IFS model, for example, consists of approximately 6.5 million points [14]. As a result, the individual fields stored within each message are about 12 megabytes (MB) in size, with a complete forecast over all 145 time steps for a single parameter adding up to over a gigabyte (GB).

[1] https://github.com/ecmwf/polytope
[2] https://github.com/ecmwf/gribjump

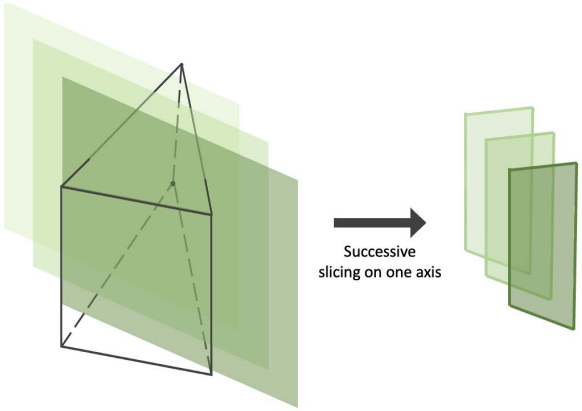


Figure 1: Polytope slicing mechanism, as illustrated in [13].

2.2 Polytope

Existing data access methods on the FDB only allow the retrieval of complete fields of data, requiring all of the data in those fields to be read and then returned to the user. As field sizes continue to grow due to increasing forecast resolution, it becomes more challenging to perform this operation for many users at a time as part of a time-sensitive workflow. To help mitigate such challenges, novel data extraction tools are needed.

One such tool is the recently developed Polytope feature extraction algorithm [13], designed to efficiently intersect arbitrary multi-dimensional polytopes with an iso-latitude datacube. The algorithm performs a series of recursive "slicing" steps, gradually reducing an n-dimensional polytope into a set of 1-dimensional points. At each step, several successive n-dimensional slices of the polytope are taken along one of the datacube's axes, as pictured in Figure 1. This process is then repeated for each axis until all points within the queried polytope are identified. Throughout the slicing process, a result tree is iteratively constructed to record all of the points found by the algorithm. This tree also provides guidance on the remaining slicing operations to be performed, detailing which lower-dimensional polytopes should intersect with which axes next.

The slicing algorithm itself does not directly extract data from the datacube however. It merely calculates the coordinates of the data points to be retrieved. For actual data extraction, the algorithm must be integrated with a compatible datacube backend that can both provide information on the available data within the datacube as well as support byte-level access to retrieve only the relevant data points.

2.3 GribJump

There is currently no well-established way to access individual bytes within fields stored on an FDB. To address this limitation, we introduce a new software tool, GribJump, designed to enable such byte-level access within the FDB. GribJump allows us to extract only the bytes found to be in the user-specified input shapes by the Polytope slicing algorithm instead of whole GRIB fields.

The core functionality of GribJump comes from its `extract` method. While data is written to the FDB after a model run, GribJump first

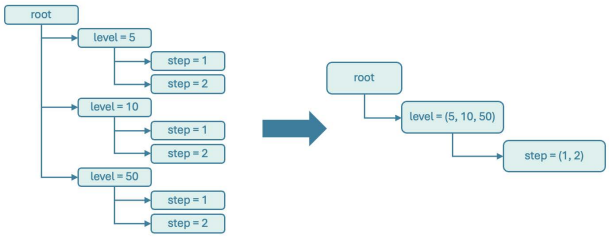


Figure 2: Tree compression example.

pre-computes an index on this data. This index is then used by the extract function to efficiently decode and locate individual bytes within fields stored in the FDB. Both simple packed data [4] and CCSDS compressed data [26] are currently supported by this mechanism.

For faster data access, GribJump has been highly optimized, particularly through parallelisation, allowing data retrieval tasks to run on multiple threads simultaneously based on the objects being accessed.

2.4 Polytope Tree Compression

As users request larger geometric shapes, the tree constructed during the Polytope slicing process can rapidly expand, resulting in a very broad structure. To address this and simplify the resulting tree, we propose a custom tree compression method, illustrated in Figure 2. This compression not only allows for a more compact representation of the results, but it is also used by the algorithm to avoid performing duplicate n-dimensional slices.

Note that not all axes of the datacube can be compressed however. In particular, compression can only be applied to axes that are defined as part of an orthogonal request shape. Due to how the tree is constructed within the Polytope algorithm, axes associated with unions can currently also not be compressed.

For meteorological data stored in an FDB, further constraints on the axis compression arise depending on the grid on which the data is stored. For most iso-latitude grids supported by Polytope, the latitude axis cannot be compressed as subsequent longitude values vary with the latitude values. The longitude axis, however, can always be compressed as it is the final axis stored on the FDB.

2.5 System Overview

In the remainder of this paper, we investigate the performance and scalability of the Polytope algorithm on data stored in an FDB. To support this analysis, we now first provide a brief overview of how both components of the system, Polytope and the FDB (via GribJump), operate together.

Initially, Polytope queries the FDB through GribJump's `axes` call to create an abstract representation of the datacube available in the FDB. This precomputation step occurs once, prior to any feature extraction, and is reusable for multiple requests as long as the data in the FDB remains unchanged.

Following this precomputation, the algorithm enters a dynamic, online phase. Here, Polytope takes a user-defined polytope, applies the slicing algorithm to the abstract datacube, and generates a result tree. This result tree, initially devoid of data, is then populated by querying the FDB through GribJump's `extract` call, after which the fully populated tree is returned to the user.

Figure 3 illustrates these interactions between Polytope and GribJump. This system is offered as part of the Polytope service at ECMWF and will also be available to users of the Destination Earth initiative through the earthkit software environment [18]. Note that this system is specific to data stored in GRIB format due to the fact that GribJump currently only supports byte access from the FDB in this data format. For datasets stored in other formats, such as climate datasets stored in netCDF [17], the Polytope algorithm can still be used to find points contained in given shapes, although specialised single byte access mechanisms will need to be implemented in order to provide an equivalent service to the system described above.

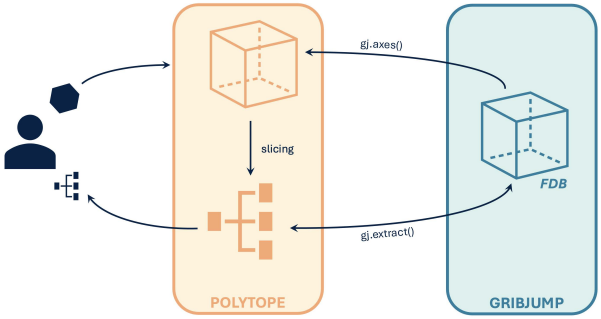


Figure 3: Overview of the Polytope and GribJump system.

3 PERFORMANCE AND SCALABILITY OF THE POLYTOPE SYSTEM

In this section, we analyse the entire Polytope system's performance and scalability. We start by examining the performance of the precomputation step of the algorithm. Note that, unlike the precomputation step, the algorithm's online phase can not be reused for multiple requests and it thus significantly affects the overall performance of the system. The main emphasis of this section is therefore on the online phase. To evaluate the performance of this step, we identify three key metrics: the slicing time, the tree construction time, and the GribJump extraction time. We then investigate how these quantities vary with the query shape to the algorithm.

3.1 Pre-computation phase

The performance of Polytope's pre-computation phase is tied to the performance of the FDB `axes` call, which itself depends heavily on how data is organised within the FDB according to its schema [3, 21, 22]. As more data on the FDB is exposed for Polytope to slice on, the cost of the FDB `axes` call will gradually increase. To access all of the current ECMWF operational data available on the FDB for example, this call takes about 4 seconds and needs to be repeated every 6 hours when new data becomes available. For larger climate datasets, calling `axes` might take longer, but the data is more

static so this cost can be paid less often as the pre-computation phase does not need to be performed as frequently. The data in the Destination Earth Climate digital twin for example is generated once and, while calling `axes` on this data takes close to 30 seconds, the output of this call can easily be cached in memory to be re-used when needed as no new data is produced in this dataset.

3.2 Online phase

In the online phase of the algorithm, three main statistics need to be considered: the slicing time, the tree construction time, and the GribJump extraction time. These metrics are influenced by various factors, ranging from technical aspects, such as the efficiency of GribJump's direct byte access to the FDB, to algorithmic factors like the dimensionality of the requested shape, how the shape is defined by the user, and the number of vertices specifying the shape.
To analyse these influences, we structure our discussion into two parts, distinguishing between technical and algorithmic impacts on the overall system performance. Finally, we conclude this section by illustrating how these combined factors affect extraction times for selected meteorological features.

3.2.1 Technical considerations. The primary technical considerations in the Polytope system stem from the GribJump component of the workflow, which directly interacts with the FDB to access data. Two critical questions arise at this stage. The first one being, how long does it take to request multiple data points from a single field, and the second one being, how does the extraction time then scale when accessing data across multiple fields.
Figure 4 addresses both of these questions. In Figure 4, in red, we observe the initial cost associated with reading and accessing data from a field. Once a field has been accessed however, we see that extracting additional points from the same field has a limited impact

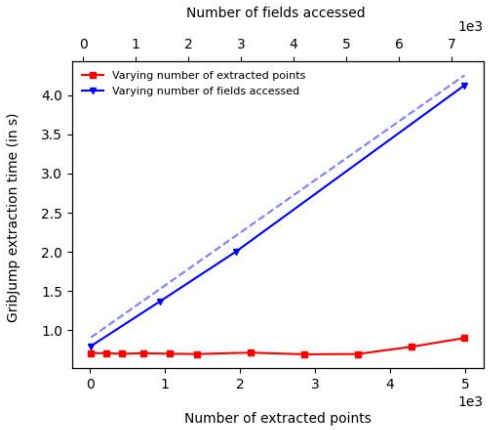


Figure 4: In red, GribJump extraction times for an increasing number of extracted points, where all points are extracted from the same field. In blue, GribJump extraction times for an increasing number of fields accessed, where a single point is extracted from each field and the dashed line shows the linear scaling behaviour of this extraction time.

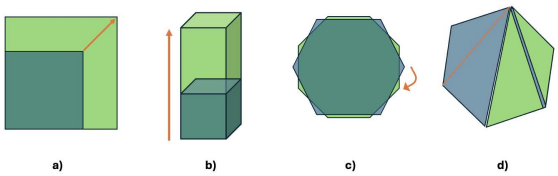


Figure 5: Schemas of the performance experiments.

on the extraction time. The blue lines in Figure 4 then show that the GribJump extraction time scales linearly with the number of fields accessed. We therefore conclude that the GribJump extraction time is mostly influenced by the number of distinct fields accessed, with minimal additional costs incurred when more points are accessed from each field.

3.2.2 Algorithmic considerations. As a computational geometry algorithm, the performance of Polytope is highly influenced by the characteristics of its query shape. In the following, we explore these dependencies in detail through various case studies, first for orthogonal boxes and then for non-orthogonal queries. For clarity, we provide a set of schemas showcasing some of the experiments we perform in Figure 5.

Orthogonal extractions. We begin by focusing our attention on orthogonal box extractions and how they scale in different dimensions.
We first investigate the effect of the shape's 2-dimensional area in Figure 6. In this figure, we extract increasingly large 2-dimensional

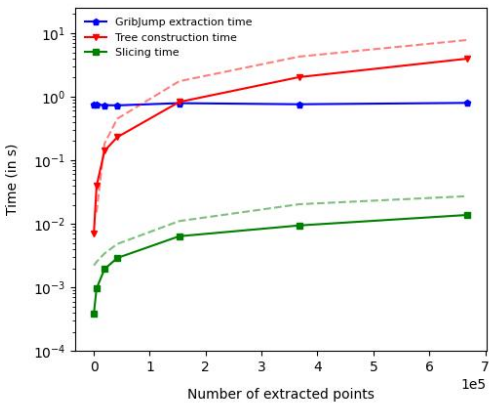


Figure 6: Polytope and GribJump extraction times for increasing 2D boxes (Figure 5a). The dashed red line represents the expected linear behaviour of the tree construction time, whilst the dashed green line represents the expected square root behaviour of the slicing time. The dashed lines are multiplied by a constant factor to enhance visibility.

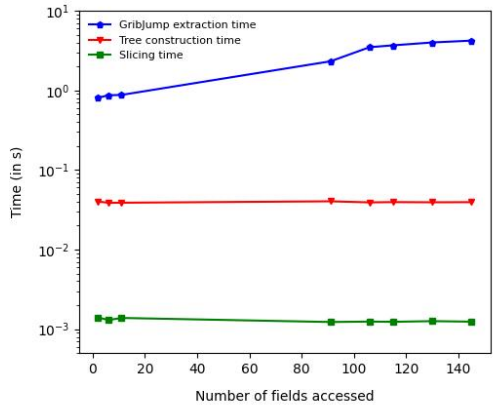


Figure 7: Polytope and GribJump extraction times for increasing 3D boxes (Figure 5b).

squares in latitude-longitude space. Each box is constructed by moving the top corner of the square further away from its lower corner. In Figure 6, in red, we observe a linear relationship between the increasing area and the tree construction time. This figure actually shows that most of the extraction time is spent constructing the result tree, with the slicing time only representing a small fraction of the overall extraction time. Note here that the tree construction time is relatively large as it not only includes the construction of the nodes of the tree, but also all of the time spent within the algorithm to set up subsequent slices. In particular, a large amount of time is spent ordering longitude values as we successively add them to the latitude tree nodes. This causes the observed linear behaviour of the tree construction time. In contrast, as we see in green in the figure, the slicing time scales with the square root of the number of extracted points. This is due to the fact that, as many meteorological grids are not iso-longitude, we do not compress the latitude axis in the extraction algorithm. We thus only perform slices for each latitude value found within the shape, which, because the sliced shapes are squares, approximately corresponds to the square root of the number of extracted points. As established before however, the extraction time is driven by the tree construction time and thus also scales linearly with the area here. Interestingly, the GribJump extraction time stays constant throughout the plot and is not impacted by the growing area of the square. This validates our earlier findings, as all points accessed here stem from the same field and thus most of the GribJump extraction cost comes from the initial object look up.

Next, we investigate the impact of slicing across multiple fields. In Figure 7, we extract 3-dimensional boxes in latitude, longitude and step. We construct these boxes by keeping the base latitude-longitude square constant, while varying the step extent of the boxes. As expected, we see in this figure that the GribJump extraction time gradually increases as we extract data from more fields. We however observe that neither the slicing nor the tree construction times vary significantly with the number of fields. This is due to the fact that we compress the boxes extracted here along the step axis, only slicing along this axis once. The resulting 2-dimensional latitude-longitude squares are the same for every box and the slicing and tree construction times are therefore not impacted by the step extension of the boxes. Figure 7 is thus a critical reminder that, as we move to higher-dimensional shapes, the performance of the system not only depends on the extracted volume, but also, perhaps more importantly, on how data is laid out in the FDB over various fields. This can further be witnessed in Figure 8, and Figure 8(a) in particular, where we scale the box extraction to higher dimensions, still keeping the base latitude-longitude square constant.

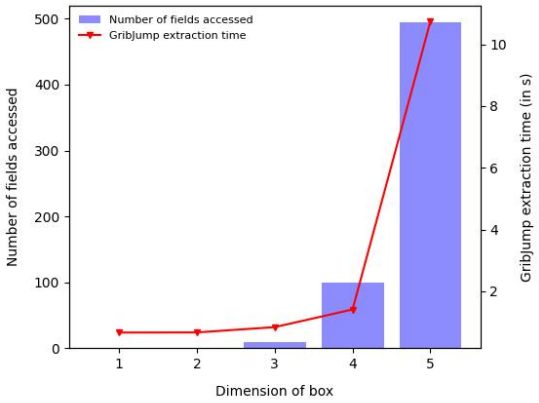
To understand how higher-dimensional slices impact the slicing time, we now extract boxes with increasing dimensions in Figure 8. We construct these boxes by successively taking the tensor product of the previous box with a range in a new dimension. In Figure 8(b), we show the number of n-dimensional slices performed for each dimension as well as the slicing time of each box. We then scale this total slicing time by the number of slices to estimate the slicing time of each individual n-dimensional slice. As we observe in the figure, 1- and 2-dimensional slices are relatively inexpensive, with slices in 4 and 5 dimensions becoming relatively slow to perform. The jump in costs between slices is due to the use of convex hulls in our slicing step, which gradually scales worse as the dimension increases. Moreover, since the latitude axis is not compressed, we need to perform all of the 2-dimensional slices along each latitude value when we move to higher-dimensional shapes. This explains the jump in both the number of slices and the total slicing time that we observe in Figure 8(b) as we extract a 2-dimensional box.

Axis compression in the request tree, as shown in Figure 2, has already been mentioned a few times throughout this section. To gain a better understanding of it, we investigate how this compression affects the different extraction times. In Figure 9, we extract boxes of increasing sizes while varying which of their axes are compressed. This shows that compression can significantly affect the performance of the slicing algorithm. In particular, we note that, as less axes are compressed, more nodes need to be created and the request tree becomes wider. This eventually results in a faster linear increase of the slicing time with the number of extracted points, which does not scale well to high-dimensional extractions.

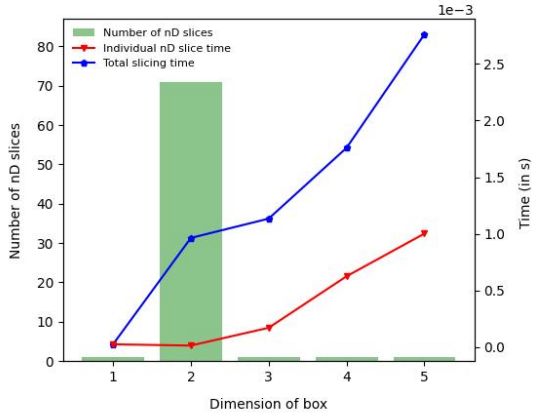
Non-orthogonal extractions. We now shift focus to explore how various shape properties affect the performance of non-orthogonal extractions.

We first analyse the effect of extracting very detailed polygons with many defining vertices. In Figure 10, we extract different polygons with an increasing number of vertices. As these polygons are constructed by approximating the same circle, they all have a very similar area so the extraction times should not change significantly. In particular, we see that the GribJump extraction time stays constant throughout, as expected. More interestingly, we observe that the slicing time of the algorithm grows with the square of the number of vertices. This can be explained by how we perform slices in the algorithm, taking the intersection of a plane with each line defined between pairs of vertices on either side of this plane.

Finally, we study the impact of constructive geometry operations on the slicing algorithm. In particular, we focus on the union operation here. In Figure 11, we extract the same 1024-sided polygon, but specified as the union of a growing set of sub-polygons. We



(a) GribJump extraction time and number of fields accessed for increasing n-dimensional boxes.



(b) Extraction times and number of nD slices performed for increasing n-dimensional boxes.

Figure 8: nD box extractions.

immediately notice that the slicing time increases linearly with the number of unions defining the shape. This is directly tied to the linear relationship between the number of unions and the number of slices to be performed, which we also observe in this figure. The linear behaviour is expected as we treat each sub-polygon in the union separately in the algorithm. We thus need to slice each sub-polygon separately, leading to a proportional increase in the number of slices.

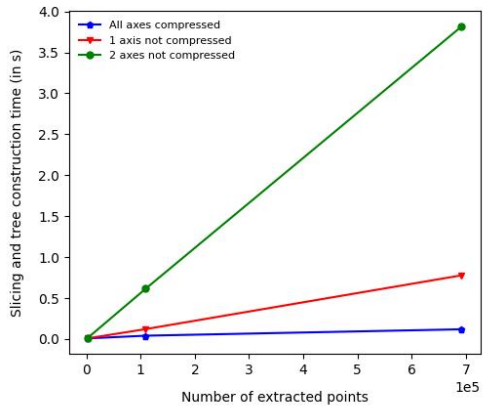


Figure 9: Extraction times of a 3D box with various levels of axis compression. The slicing and tree construction time is shown in blue for when all axes in the box are compressed, in red when the longitude axis is not compressed and in green when both the longitude and step axes are not compressed.

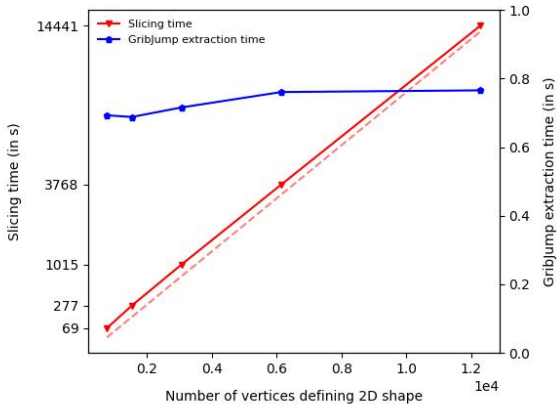


Figure 10: Extraction times of polygons of same area but with an increasing number of defining vertices (Figure 5c). Note that the slicing time axis appears in square root scale with the dashed red line representing the expected squared behaviour of the slicing time.

3.2.3 Extracting meteorological features. In real-world scenarios, the factors described above are combined to influence Polytope's total response time. To assess the algorithm's efficiency for meteorological applications, we examine several examples.
In Figure 12, we analyse data extraction times for different countries, identifying where time is spent. Countries are specified as unions of convex polygons with each polygon being treated independently by the algorithm and no compression of the involved axes. As expected, this impacts the slicing and tree construction times, which are more consequent for larger shapes, such as France

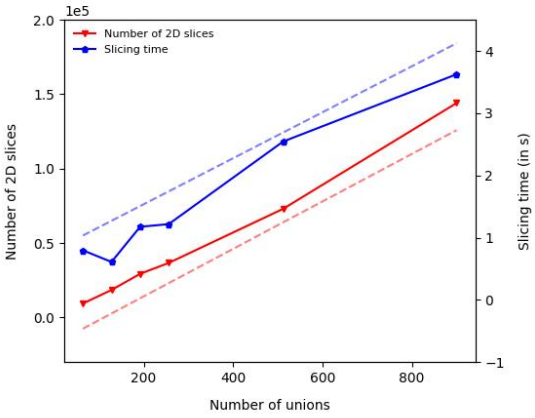


Figure 11: Slicing time and number of 2D slices for a polygon specified as the union of an increasing set of sub-polygons (Figure 5d). The dashed lines represent the corresponding expected linear behaviour of these quantities.

when compared to Switzerland, but also for shapes that have more irregularities, such as the UK when compared to the more rectangular Germany. This is further emphasized in Figure 12(b), where we show the stark contrast between the smaller and less-detailed countries in Europe and the large, more complex shape of Canada, which is exponentially more expensive to extract. Note that the fact that countries are defined as unions currently also affects the GribJump extraction time, which is quite considerable here for a single field extraction. This is due to the fact that each sub-polygon is treated separately as a different request to extract from GribJump and could be optimised in the future.

These examples focus on 2D extractions in latitude-longitude space, so the impact of accessing multiple data fields cannot be observed from this figure. Instead, in Figure 13(a), we address this point, showing a point timeseries extraction across multiple fields. Here, the slicing time does not vary and remains negligible, while the GribJump extraction time significantly increases as more fields are accessed, as expected. Finally, in Figure 13(b), we present an example timeseries extraction over the UK, where both more data points and fields are accessed. In this case, both slicing and GribJump extraction times increase, with the latter scaling linearly with the number of fields accessed.

In all examples, Polytope's extraction times were at most a few seconds, even for more complex cases, with the Canada retrieval taking under a minute. These results suggest that the slicing algorithm is an efficient data extraction method for various meteorological features.

4 I/O AND DATA REDUCTIONS

The main aim of the Polytope feature extraction algorithm is to improve access to large-scale meteorological data. A key aspect of this objective is to minimise the I/O load in operational weather systems by limiting data transfer to only what is essential for specific applications.

Figure 14 illustrates the data reduction achieved with Polytope compared to extracting a bounding box around several example countries. We observe a significant reduction of up to almost an order of magnitude for non-rectangular countries such as Italy. Even for more box-shaped countries like Germany, the reduction remains clearly visible on the graph in logarithmic scale. Note that the examples in the figure are of 2-dimensional extractions. As more fields are accessed however, the potential for data reduction can significantly increase. For high-dimensional shapes, such as spatio-temporal trajectories in particular, data reduction can easily exceed 99%.

This data reduction can then directly be translated into a faster access to data, especially for data spanning over multiple fields. For example, retrieving the point time series from above using Polytope took less than a second, compared to several seconds when retrieving the full fields from the FDB. For the point time series over all ensemble numbers, Polytope completed the extraction in about 4 seconds, while accessing the entire fields directly from the FDB took nearly 2 minutes.

Such improvements in data reduction and performance enable more efficient data access for users. By minimising data transfer between operational systems and their users, Polytope helps reduce the I/O load incurred per user on the storage system. This results in a more efficient data access pattern which can support a large number of users simultaneously.

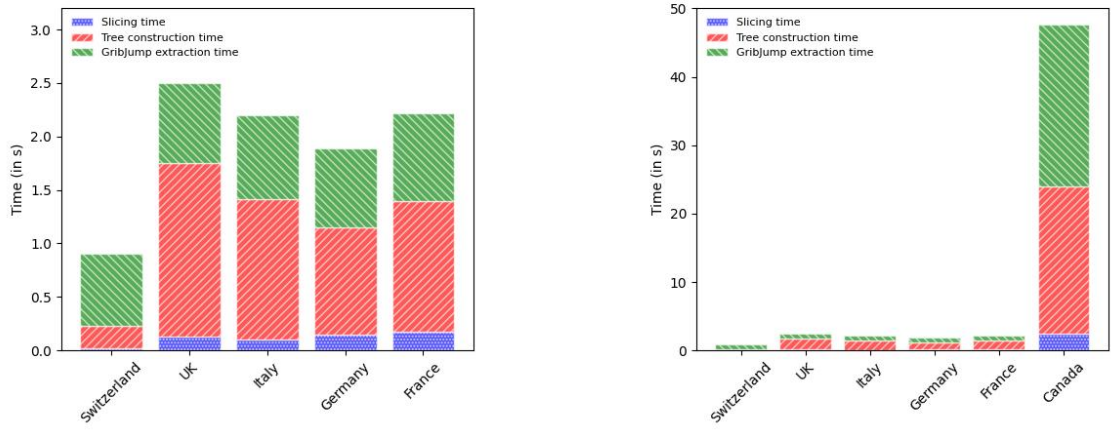
5 DISCUSSION AND FUTURE WORK

In the current context of open data initiatives such as Destination Earth, the Polytope feature extraction algorithm provides a reliable solution to a rising demand in meteorological data access. By limiting the amount of data read and transferred to users to the bare minimum, it represents an efficient and scalable alternative to traditional data extraction techniques. Beyond data retrieval, Polytope also introduces the possibility for interactive workflows as access to data in near real-time starts to become feasible. This paves the way for more responsive applications in the fields of meteorology and climate change.

While Polytope provides substantial benefits to operational weather forecasting systems however, the feature extraction algorithm still has a few notable limitations.

The first challenge we identified in this paper was in the pre-computation phase of the algorithm. In this part of the Polytope system, we experience a trade-off between speed and the range of shapes that can be extracted. To address this, one approach is to integrate the pre-computation phase within the online phase. In particular, we plan to optimise the FDB `axes` call to be called recursively as we build the request tree in future implementations of the slicing algorithm.

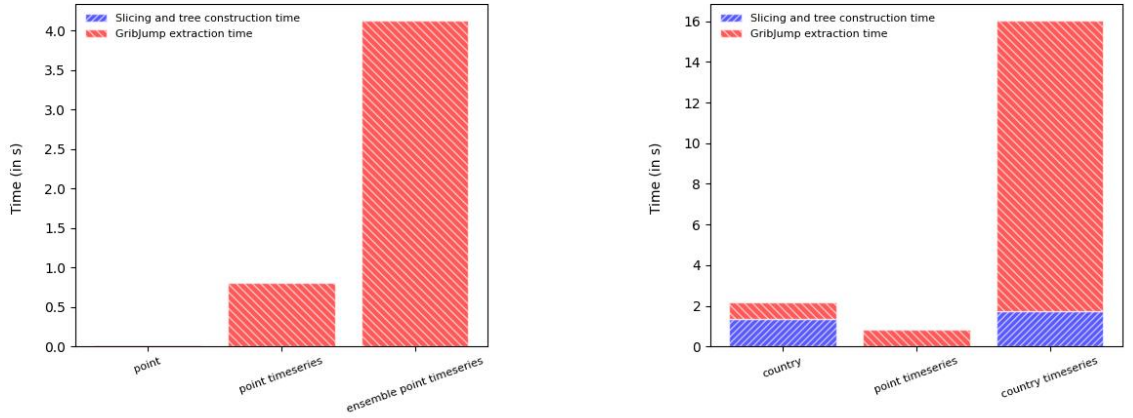
A second challenge that we identified during our performance analysis is that polygons with a large number of vertices are difficult to slice, and their extraction can quickly become computationally intractable. To overcome this, line-simplification methods [2], such as the Douglas-Peucker algorithm [9], can be used to approximate highly detailed polygons by polygons with fewer vertices. However,



(a) European countries

(b) Comparison between European countries and Canada

Figure 12: Extraction times for different country shapes.



(a) Point timeseries

(b) Country timeseries

Figure 13: Extraction times for different timeseries.

for Polytope to ensure that all points within a requested shape are returned to users, the line reduction algorithm that we use must meet an additional requirement. It should consistently overestimate the polygon boundaries to capture the entire shape, whilst still preserving the overall shape of the polygon in order to avoid unnecessary data extraction. As, to the best of our knowledge, there is no existing algorithm which fully meets this criteria, a key area of future work will involve developing and implementing a suitable polygon approximation technique specifically for this purpose.
Additional improvements to the feature extraction algorithm include optimising the tree compression and refining the interface between Polytope and GribJump. Our analysis indicated that tree compression significantly affects overall system performance. Extending this compression to handle more axes and complex shapes will thus be a crucial step toward increasing Polytope's scalability. Enabling GribJump to then work directly with this tree structure will also help reduce the current overhead from translating between different object types within the Polytope-GribJump interface. Furthermore, most of the slicing and tree construction in Polytope is currently implemented in Python and, due to the nature of the algorithm, existing compiled backends such as numpy [23] do not provide an advantage for performance. Significant speed-ups could instead be achieved by writing portions of Polytope in a more performant language, such as C++ or Rust [15].
In the longer term, Polytope's adaptability to different data sources

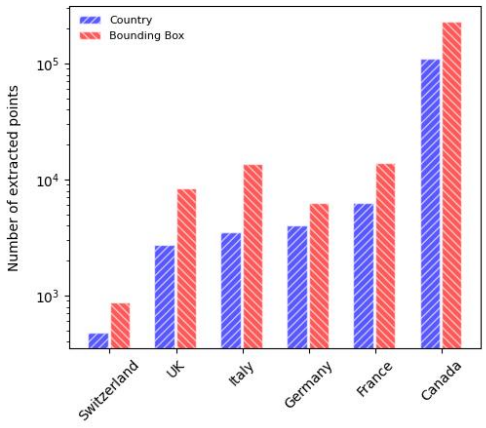


Figure 14: Number of extracted points for different country shapes and their bounding boxes. Note that the number of extracted points is shown in log scale.

is another important area for development. This includes both an assessment of necessary changes to GribJump for efficient data retrieval over networks instead of local file systems, as well as the development of alternative slicing algorithms for data stored on unstructured grids. These changes are particularly relevant for the Destination Earth initiative, where data on the data bridge is transferred over a network and certain datasets are stored on icosahedral and Lambert grids.

ACKNOWLEDGMENTS

The work presented in this paper has been produced in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

REFERENCES

[1] Karamat Ali, Roshan M Bajracharyar, and Nani Raut. 2017. Advances and challenges in flash flood risk assessment: A review. *Journal of Geography & Natural Disasters* 7, 2 (2017), 1–6.
[2] Robert G Cromley. 1991. Hierarchical methods of line simplification. *Cartography and Geographic Information Systems* 18, 2 (1991), 125–131.
[3] ECMWF. 2017. FDB Documentation. https://fields-database.readthedocs.io/en/latest/ Accessed on 2024-11-12.
[4] ECMWF. 2024. GRIB2 Regulations. https://codes.ecmwf.int/grib/format/grib2/regulations/ Accessed on 2024-11-12.
[5] ECMWF. 2024. Key facts and figures. https://www.ecmwf.int/en/about/media-centre/key-facts-and-figures Accessed on 2024-11-12.
[6] Sandro Fiore, Alessandro D'Anca, Donatello Elia, Cosimo Palazzo, Dean Williams, Ian Foster, and Giovanni Aloisio. 2014. Ophidia: a full software stack for scientific data analytics. In *2014 international conference on high performance computing & simulation (HPCS)*. IEEE, 343–350.
[7] Thomas Geenen, Nils Wedi, Sebastian Milinski, Ioan Hadade, Balthasar Reuter, Simon Smart, James Hawkes, Emma Kuwertz, Tiago Quintino, Emanuele Danovaro, et al. 2024. Digital twins, the journey of an operational weather system into the heart of Destination Earth. *Procedia Computer Science* 240 (2024), 99–108.
[8] Luke Gosink, John Shalf, Kurt Stockinger, Kesheng Wu, and Wes Bethel. 2006. HDF5-FastQuery: Accelerating complex queries on HDF datasets using fast bitmap indices. In *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*. IEEE, 149–158.
[9] John Edward Hershberger and Jack Snoeyink. 1992. Speeding up the Douglas-Peucker line-simplification algorithm. (1992).
[10] Jörn Hoffmann, Peter Bauer, Irina Sandu, Nils Wedi, Thomas Geenen, and Daniel Thiemert. 2023. Destination Earth–A digital twin in support of climate services.
[11] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. 2023. Learning skillful medium-range global weather forecasting. *Science* 382, 6677 (2023), 1416–1421.
[12] Simon Lang, Mihai Alexe, Matthew Chantry, Jesper Dramsch, Florian Pinault, Baudouin Raoult, Mariana CA Clare, Christian Lessig, Michael Maier-Gerber, Linus Magnusson, et al. 2024. AIFS-ECMWF's data-driven forecasting system. *arXiv preprint arXiv:2406.01465* (2024).
[13] Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, and Tiago Quintino. 2023. Polytope: An Algorithm for Efficient Feature Extraction on Hypercubes. *arXiv preprint arXiv:2306.11553* (2023).
[14] Sylvie Malardel, Nils Wedi, Willem Deconinck, Michail Diamantakis, Christian Kühnlein, George Mozdzynski, Mats Hamrud, and Piotr Smolarkiewicz. 2016. A new grid for the IFS. *ECMWF newsletter* 146, 23-28 (2016), 321.
[15] Nicholas D Matsakis and Felix S Klock. 2014. The rust language. *ACM SIGAda Ada Letters* 34, 3 (2014), 103–104.
[16] Anders Persson and Federico Grazzini. 2007. User guide to ECMWF forecast products. *Meteorological Bulletin* 3, 2 (2007).
[17] Russ Rew and Glenn Davis. 1990. NetCDF: an interface for scientific data access. *IEEE computer graphics and applications* 10, 4 (1990), 76–82.
[18] Iain Russell, Tiago Quintino, Baudouin Raoult, Sándor Kertész, Pedro Maciel, James Varndell, Corentin Carton de Wiart, Edward Comyn-Platt, Olivier Iffrig, James Hawkes, et al. 2024. Introducing earthkit. https://www.ecmwf.int/en/newsletter/179/computing/introducing-earthkit
[19] Christoph Schär, Oliver Fuhrer, Andrea Arteaga, Nikolina Ban, Christophe Charpilloz, Salvatore Di Girolamo, Laureline Hentgen, Torsten Hoefler, Xavier Lapillonne, David Leutwyler, et al. 2020. Kilometer-scale climate models: Prospects and challenges. *Bulletin of the American Meteorological Society* 101, 5 (2020), E567–E587.
[20] Mark Schelbergen, Peter C Kalverla, Roland Schmehl, and Simon J Watson. 2020. Clustering wind profile shapes to estimate airborne wind energy production. *Wind Energy Science Discussions* 2020 (2020), 1–34.
[21] Simon D Smart, Tiago Quintino, and Baudouin Raoult. 2017. A scalable object store for meteorological and climate data. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. 1–8.
[22] Simon D Smart, Tiago Quintino, and Baudouin Raoult. 2019. A high-performance distributed object-store for exascale numerical weather prediction and climate. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. 1–11.
[23] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in science & engineering* 13, 2 (2011), 22–30.
[24] Nils Wedi, Peter Bauer, Irina Sandu, Jörn Hoffmann, Sophia Sheridan, Rafael Cereceda, Tiago Quintino, Daniel Thiemert, and Thomas Geenen. 2022. Destination earth: High-performance computing for weather and climate. *Computing in Science & Engineering* 24, 6 (2022), 29–37.
[25] World Meteorological Organization (WMO). 2023. Manual on Codes, Volume I.2 – International Codes. https://library.wmo.int/idurl/4/35625
[26] Pen-Shu Yeh, Philippe Armbruster, Aaron Kiely, Bart Masschelein, Gilles Moury, Christoph Schaefer, and Carole Thiebaut. 2005. The new CCSDS image compression recommendation. In *2005 IEEE Aerospace Conference*. IEEE, 4138–4145.
[27] Xu Zhou, Kun Yang, Lin Ouyang, Yan Wang, Yaozhi Jiang, Xin Li, Deliang Chen, and Andreas Prein. 2021. Added value of kilometer-scale modeling over the third pole region: a CORDEX-CPTP pilot study. *Climate Dynamics* (2021), 1–15.

4 Weather and Climate Digital Twins Use Case

We have now designed a complete system for feature extraction from NWP data stores. In the following paper [71], we demonstrate the applicability of this system to the Destination Earth digital twins. We first describe several custom capabilities of the Polytope software that enable efficient extraction in this context, including support for cyclic axes and multiple types of iso-latitude grids. We then explain how the Polytope feature extraction service has been implemented to serve data from the Destination Earth initiative and illustrate its operation through a point time-series extraction from the Climate Digital Twin.

This work was published in the 2024 proceedings of the International Astronautical Congress (IAC). I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *J. Hawkes*: Conceptualisation, Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

75th International Astronautical Congress (IAC), Milan, Italy, 14-18 October 2024.

IAC–24–B1.IP.103.x81121

Polytope: Extracting Features from Large-Scale Datacubes

Mathilde Leuridan[a]*, James Hawkes[b], Tiago Quintino[c]

[a] *ECMWF, Germany*, Mathilde.Leuridan@ecmwf.int
[b] *ECMWF, UK*, James.Hawkes@ecmwf.int
[c] *ECMWF, UK*, Tiago.Quintino@ecmwf.int
* Corresponding Author

Abstract

As part of the EU's ambitious Destination Earth initiative, the European Centre for Medium Range Weather Forecasts (ECMWF) is building the Digital Twin Engine, a new software infrastructure designed to efficiently handle the petabytes of data produced daily by large-scale earth system digital twins. One important component of the Digital Twin Engine is the Polytope software. This software implements ECMWF's novel feature extraction algorithm, which uses concepts of higher-dimensional computational geometry to efficiently retrieve arbitrary user-specified shapes from high-dimensional datacubes. When this task is performed server-side, in-situ with the data, Polytope can reduce data sizes by several orders of magnitude and return tailored analysis-ready data to users. We combine this feature extraction mechanism with the capability to directly jump to specified byte ranges in a data store, which thus allows us to minimise I/O in the system to the bare minimum required to serve user-specified requests. In this paper, we describe the Polytope software and its latest developments before highlighting how this software can be used to address the challenges related to growing data sizes faced by the Earth Observation and Weather Forecasting communities in large-scale projects such as the Destination Earth initiative.

Keywords: Datacubes, Feature Extraction, Data Infrastructure, Data Management

1. Introduction

Over the past few years, the field of weather forecasting has grown incredibly fast and seen many revolutionary changes. From higher resolution models, such as in the European Union's Destination Earth initiative [10], to the introduction of machine learning (ML) based weather forecasts, such as the GraphCast model [4], these developments all hold significant scientific value. Although these are major advances in the field however, they do not come without technical challenges, especially in terms of data management.

One of the main technical challenges is the growing volume of data produced each day by weather models. An increase in spatial resolution of a weather model by a factor 2 results in a 4-fold increase in model data output. With higher resolution models, scientists have thus been faced with the difficult task of handling these rapidly growing data volumes. In the Destination Earth initiative, for example, the digital twins will produce over a petabyte of data per model run. Meanwhile, the rise of ML models for weather forecasting has significantly lowered the computational cost associated with running simulations of the weather. Where traditional models require hours of computation time, ML models can run within a few minutes once they have been trained. It is thus possible to run these weather models much more frequently than the standard 2 to 4 simulations a day. This only further increases the volume of data produced each day.

When considering the combined challenge of larger model outputs with the increased frequency of model runs, it is becoming essential for scientists to develop

new technologies which can efficiently handle the large volume of produced model data, especially if the entirety of this data needs to remain easily accessible and available for scientific purposes.
It is with such challenges in mind that the European Centre for Medium Range Weather Forecasts (ECMWF) is developing the Digital Twin Engine (DTE) as part of the Destination Earth initiative. Within the DTE, the Polytope software will serve as the main data access mechanism for digital twin data. In this paper, we first introduce the Polytope concept and some of its particular features which make it a useful tool for weather forecasting applications, before discussing how it fits within the data infrastructure at ECMWF and how it will be used as part of the DTE in the Destination Earth initiative.

2. Polytope Concept

Polytope is ECMWF's new data extraction service. Unlike traditional extraction techniques, which only support the extraction of axis-aligned "box" shapes, Polytope is instead designed to extract polytopes, or in other words multidimensional polygons, from datacubes. We now explain the concept behind the Polytope approach as well as some of its main benefits. This section is based on [5], where more details about the Polyope algorithm can be found.

2.1 Traditional Extraction Methods

As mentioned before, most current data extraction techniques only allow users to extract orthogonal box shapes from datacubes. Using these methods, users specify ranges to extract along each of the datacubes' axes and get back all the data contained within the box they have defined.
For many applications however, this is not efficient. Indeed, for a lot of scientific use cases, scientists are interested in specific sub-regions of their collected data, which are not necessarily box-shaped. Using current methods, they would have to retrieve a bounding box of their non-orthogonal shape, containing much more data than they actually need, and then extract the useful information themselves.
Consider for example a meteorologist interested in temperature data over a country such as Italy. With traditional data extraction techniques, this meteorologist would have to extract temperature data over a rectangle surrounding Italy, thus getting back a lot more data than required for their application.

2.2 Polytope Extraction Algorithm

As an alternative to the traditional extraction methods mentioned above, ECMWF has implemented the Polytope algorithm. Polytope was specifically designed to extract non-orthogonal shapes from datacubes. By using tools from higher-dimensional computational geometry, it is able to extract polytope stencils from datacubes, as pictured in Figure 1.

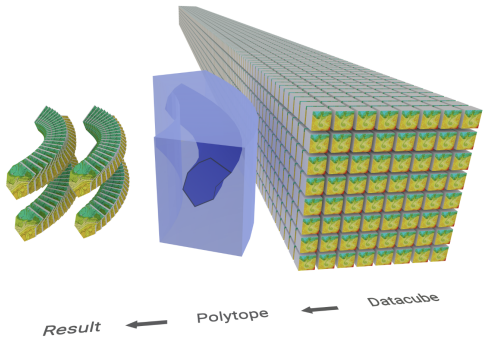


Fig. 1. Concept of the Polytope algorithm. Taken from [5]

The algorithm computes the bytes that are contained within a user-specified shape by successively slicing this shape as described in [5]. We refer to this process as the slicing algorithm. As shown in Figure 2, if we then combine this with another tool that has the capability to only read specific bytes in a data store, it then becomes possible to return only the data of interest to the user. Most importantly however, note that this algorithm only requires the system to read the bytes of interest to the user from the data store, instead of spending time reading more unnecessary bytes. As we highlight in the next section, this has many benefits.

2.3 Benefits

The Polytope extraction mechanism has many benefits compared to more traditional methods. As

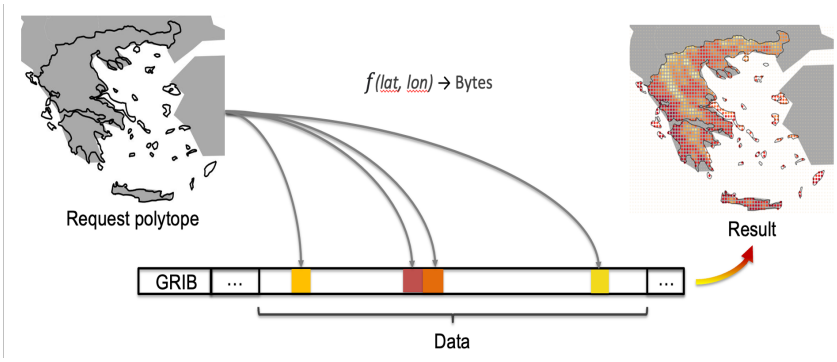


Fig. 2. Polytope feature extraction mechanism.

explained above, by using the Polytope mechanism, users only need the system to read the bytes of interest to them, instead of a lot of additional unnecessary data. This significantly reduces the I/O load on the system, which will consequently make the data retrieval system faster for all users. Moreover, since the I/O load on the system is reduced, the Polytope extraction mechanism will also allow more users to access data at the same time. As a secondary benefit, Polytope also removes the need for processing after extraction on the user's side since only the requested data is returned. Polytope thus constitutes a modern solution to the challenge of efficient data retrieval from large-scale data stores, which scales well with both increasing data volumes as well as increasing user demand.

3. Weather and Climate Data

As long as a dataset can be represented as an ordered datacube along a set of well-defined axes, Polytope can be used to extract features from this datacube. This holds for data generated in any scientific field, from medicine to Earth Observation. In this paper however, we focus on using Polytope for weather and climate data. We first examine two common data storage structures for this type of data. In particular, we explain why we have chosen these two formats to be supported as datacube backends to Polytope. We then discuss some special properties of weather and climate data which require special attention in the slicing algorithm, before describing how we handle them within Polytope.

3.1 Datacube Structures

Polytope currently supports extraction from 2 datacube structures, namely XArray data arrays [3] and FDB object stores [9]. Whereas the XArray data array is one of the most common data formats used for working with complex scientific data, such as weather and climate data amongst others, the FDB object store is a much more tailored data structure optimised especially for meteorological use cases. We now discuss the advantages of implementing both of these formats as backends to Polytope, whilst also taking into account some of the limitations associated with each of them.

XArray Data Arrays are used in many scientific applications to store data along a set of well-defined dimensions. Although they are often used for geospatial or spatio-temporal data, they remain a very generic structure which can be used to store virtually any well-structured axis-aligned data. Furthermore, many other data formats have encoders which transform these various formats into XArray data arrays. Examples include TileDB arrays [7] and Zarr data stores [6], as well as data stored in netCDF [8] and GRIB files [2]. It is thus possible to extract data from those formats through Polytope by using the XArray datacube backend. This shows that the XArray datacube backend is very versatile and makes Polytope accessible to users from a range of different fields. One significant limitation of this backend

however is that XArray data arrays can not hold data which has a more complex structure, such as irregular axis lengths or branching of the datacube. In ECMWF's weather data for example, data might be available on different timesteps according to the chosen parameter, showcasing irregularity in the data. Meanwhile, unlike pressure level data, surface data does not require the specification of an additional level axis, which shows the branching nature of the ECMWF weather datacube. Both of these irregular and branching properties prevent ECMWF operational data from being efficiently represented as an XArray data array and an alternative object is thus needed to store data exhibiting such properties.

FDB Object Stores are, in contrast to XArray data arrays, more tailored to meteorological data. They store data according to a hierarchical schema and thus provide better support for special datacube properties such as the branching or irregular data mentioned previously. At ECMWF, FDB is used both in operations and in the Destination Earth initiative to store the data produced by the weather model and digital twins respectively. Through the newly developed GribJump software [1], FDB object stores are now also byte-addressible and can efficiently return singular bytes within their stored data. This enables the Polytope feature extraction mechanism shown in Figure 2 to run directly on these FDB object stores. For ECMWF data, this implies that the data does not need to be first downloaded and stored as an XArray before being able to perform feature extraction using Polytope. Implementing FDB object stores as a datacube backend to Polytope thus allows for faster access to weather data features than the XArray data array backend does. FDB object stores however only store data in GRIB format, which is a data format specific to the weather and climate community. Because of this, the FDB datacube backend cannot be used to perform Polytope feature extraction for applications from more diverse fields, which is the main limitation of this backend.

3.2 Datacube Properties

Weather and climate data has some unique characteristics, which are mostly due to how the data is produced and stored. These characteristics affect both the structure of the associated datacube, as well as the way in which data can be queried on this datacube. To create a more seamless data extraction mechanism for Polytope users, special operations on the datacube axes, called "transformations", have been introduced in the extraction algorithm. Some of the main transformations that have been implemented within the Polytope algorithm are highlighted in the rest of this section.

Cyclic axes One specificity of weather data is that it has a cyclic longitude axis. For example, a data point at longitude 420 is equivalent to a data point at longitude 60. Users who request a point at longitude 420, will thus expect the same output as if they requested a point a longitude 60. Weather and climate datasets however usually only store data on a longitude range of [0, 360] or [-180, 180]. To allow users to seamlessly request points outside of these ranges as well as shapes which cross the cyclic boundary of the data, Polytope implements a special translation of the user-requested points to points lying within the stored longitude ranges.

Grids Weather data is usually computed on structured isolatitude grids. For each grid, the data points are then stored along a set of indices $i = 0, \ldots, N_{points}$ according to a scheme specific to that grid. To allow users to request shapes defined along the latitude-longitude axes instead of in terms of these grid indices, Polytope implements the mappings between grid indices and corresponding latitude-longitude points for several commonly used grids. Currently, Polytope supports octahedral, reduced latitude-longitude, regular latitude-longitude as well as HEALpix grids.

Merged axes At ECMWF, weather data is stored along two axes of time: the datetime that the forecast was run, and the timestep of the forecast into the future. For the former, weather data is typically addressed using a date and time combination, expressing the date of the forecast and the particular daily cycle (e.g. date=20240601, time=00). When requesting timeseries spanning several days however, separating the date and time axes like this becomes inconvenient as the user request quickly becomes more complicated to formulate. Instead, Polytope

implements a merging of these axes into just one datetime axis which allows for a more seamless user experience.

4. Polytope in Destination Earth

The European Union's Destination Earth (DestinE) initiative is a project to build digital twins of the earth [10]. Due to their fine resolution, these digital twins are predicted to produce an unprecedented volume of data, in the order of a petabyte per day. To efficiently manage and work with this large amount of data, the DestinE initiative is implementing the Digital Twin Engine (DTE), a software infrastructure especially designed to power the digital twins. Polytope is the main data access mechanism in the Digital Twin Engine, allowing its users to perform feature extraction on the digital twins' output weather and climate data. Polytope and the DTE play a critical role in the DestinE initiative as they are an integral part of ECMWF's software and services required to access any DestinE digital twin data.
In this section, we first discuss the motivation for using Polytope to access and retrieve data in the Digital Twin Engine, highlighting in particular why such a new data retrieval system is needed in the Destination Earth initiative. We then describe some of the features that can already be extracted by Polytope users, before showing an example of the feature extraction algorithm on digital twin output data.

4.1 Motivation

In weather forecasting, output data is written and stored in fields, or layers of the earth. Traditionally, data extraction mechanisms would return entire fields to users, who would then do some post-processing locally to retrieve the data which they need. With the higher resolution models in Destination Earth's digital twins, these fields are significantly growing in size. Returning entire fields to users thus implies much more data transfer than before. Returning whole fields of this size also puts significant load on the whole IO system, which reduces the number of users which can be served. Traditional data extraction mechanisms therefore do not scale well to the growing data sizes encountered in the Destination Earth initiative.
In comparison, Polytope only retrieves the specific bytes of data needed by users, reducing the data transfer in the system to the bare minimum required for user applications. Using Polytope thus makes data retrieval both efficient and scalable to a larger number of users.
Additionally, Polytope helps make retrieving digital twin data more interactive as users can request custom shapes specific to their applications. Since only the data within the user-specified shapes is retrieved, Polytope also helps reduce the data post-processing work on the users' side. This highlights the benefits of Polytope not only for the digital twin system, but also for individual users.

4.2 Supported Features

Polytope can extract any user-defined shape of data through its built-in interface. To make it more accessible to a wide range of users however, some commonly requested features have been implemented in a higher-level domain-specific interface. They currently include

- timeseries,
- vertical profiles,
- user-specified regions and
- bounding boxes.

4.3 Example

We now show an example of the Polytope feature extraction algorithm on Destination Earth's Climate Adaptation digital twin. In this example, we extract a timeseries around Lisbon in Portugal. In particular, we retrieve hourly data for an entire year so that we can observe the changes in 2 meter temperature over this period. The extracted data is shown in Figure 3. Instead of extracting global fields for all of these timesteps in the forecast, using Polytope, we retrieved only the relevant data points for our application around Lisbon. This constitutes a significant improvement as only about 1.4 MB of data were extracted instead of hundreds of gigabytes. Not only is this more efficient for the digital twin system, but it also allows the user to focus more on their application rather than worrying about downloading and

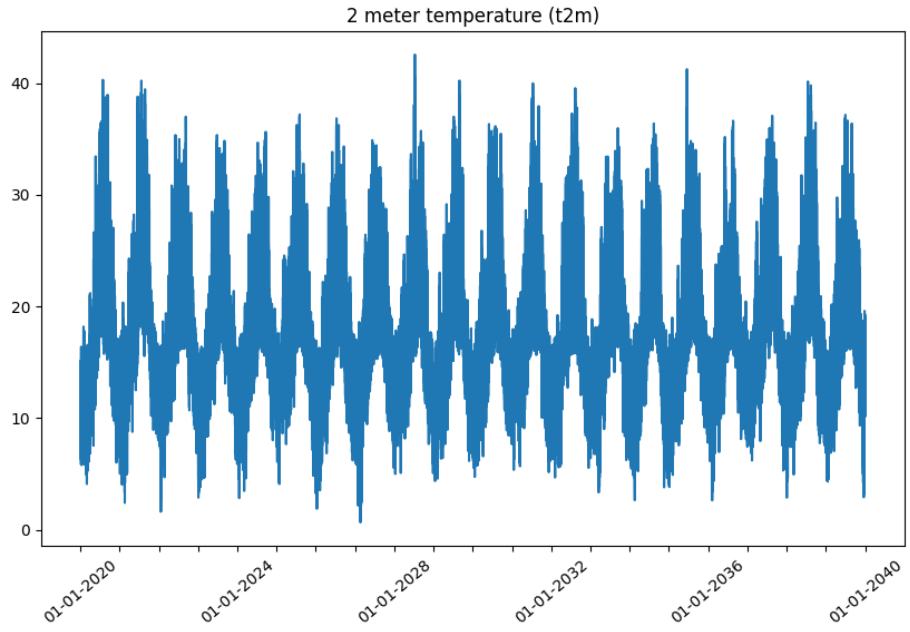


Fig. 3. Timeseries of the two meter temperature (t2m) in degrees Celsius over Lisbon, Portugal. This figure was produced using hourly t2m data from the Destination Earth climate adaptation digital twin over the period ranging from 01-01-2020T00:00:00 to 31-12-2039T23:00:00.

then processing large amounts of data on their local machine. This example thus perfectly showcases the benefits of the Polytope feature extraction algorithm for both an operational data-producing system, like the digital twin, and its users.

5. Conclusion

In this paper, we introduced the Polytope algorithm and highlighted some of its important features which make it suitable for weather forecasting applications. We then discussed its crucial role within the Destination Earth initiative as part of the Digital Twin Engine. Although currently the algorithm is only a prototype, with phase two of the Destination Earth initiative, Polytope will soon become an operational service. As we move towards this goal, future work will entail a more thorough performance analysis of the algorithm as well as subsequent optimisations. Additionally, various surrounding services will be developed to make Polytope available to a wider range of both technical and non-technical users.

6. Aknowledgments

The work presented in this paper has been produced in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Com-

mission can be held responsible for them.

References

[1] Gribjump software. https://github.com/ecmwf/gribjump.

[2] Jean Claude Bergès. Support of wmo binary format (bufr and grib). In *Proceedings of the Open source GIS-GRASS user conference, Trento, Italy*, volume 11, page 13. Citeseer, 2002.

[3] Stephan Hoyer and Joe Hamman. xarray: ND labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 2017.

[4] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. GraphCast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

[5] Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, and Tiago Quintino. Polytope: An algorithm for efficient feature extraction on hypercubes. *arXiv preprint arXiv:2306.11553*, 2023.

[6] Alistair Miles, John Kirkham, Martin Durant, James Bourbeau, Tarik Onalan, Joe Hamman, Zain Patel, shikharsg, Matthew Rocklin, raphael dussin, Vincent Schut, Elliott Sales de Andrade, Ryan Abernathey, Charles Noyes, sbalmer, pyup.io bot, Tommy Tran, Stephan Saalfeld, Justin Swaney, Josh Moore, Joe Jevnik, Jerome Kelleher, Jan Funke, George Sakkis, Chris Barnes, and Anderson Banihirwe. zarr-developers/zarr-python: v2.4.0, April 2020. doi:10.5281/zenodo.3773450.

[7] Stavros Papadopoulos, Kushal Datta, Samuel Madden, and Timothy Mattson. The tiledb array data storage manager. *Proceedings of the VLDB Endowment*, 10(4):349–360, 2016.

[8] Russ Rew and Glenn Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[9] Simon D Smart, Tiago Quintino, and Baudouin Raoult. A scalable object store for meteorological and climate data. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–8, 2017.

[10] Nils Wedi, Peter Bauer, Irina Sandu, Jörn Hoffmann, Sophia Sheridan, Rafael Cereceda, Tiago Quintino, Daniel Thiemert, and Thomas Geenen. Destination Earth: High-performance computing for weather and climate. *Computing in Science & Engineering*, 24(6):29–37, 2022.

5 Earth Observation Datacubes Use Case

Beyond weather and climate data, the Polytope feature extraction algorithm is applicable to a wider range of Earth science domains. In the following paper [70], we demonstrate its application to Earth observation (EO) satellite imagery and derived data products. We first review existing datacube implementations in the EO community, then show how Polytope can be integrated with these systems, and finally present several representative use cases. We highlight that EO data management faces challenges similar to those encountered in NWP, making the Polytope feature extraction approach broadly applicable across multiple Earth science applications.

This work was published in the 2023 proceedings of the International Astronautical Congress (IAC). I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *J. Hawkes*: Conceptualisation, Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

74th International Astronautical Congress (IAC), Baku, Azerbaijan, 2-6 October 2023.

IAC–23–B1.4.3.x75529

Polytope: Feature Extraction for Improved Access to Petabyte-Scale Earth Observation Datacubes

Mathilde Leuridan[a]*, James Hawkes[b], Tiago Quintino[c]

[a] *ECMWF, Germany*, Mathilde.Leuridan@ecmwf.int
[b] *ECMWF, UK*, James.Hawkes@ecmwf.int
[c] *ECMWF, UK*, Tiago.Quintino@ecmwf.int
* Corresponding Author

Abstract

Earth Observation (EO) has been one of the main drivers of the aerospace sector for over 60 years. With increases in satellite instrument precisions and EO satellite launches, the data produced by this field has grown exponentially in the past few years. To continue using EO data in downstream applications, it is thus essential to find new methods of representing and accessing this data, which are both efficient and intuitive. In recent years, EO data has often been represented as datacubes, with initiatives such as the Earth System Data Cube or Open Data Cube paving the way towards a unified access to different EO data streams. Indeed, by combining different observation data types into single datacubes, EO data access has become easier for users, who can now find most of the data they need for their scientific applications in one place. As these EO datacubes grow in size, there is no clear-cut way of accessing them efficiently. In this paper, we describe an algorithm (Polytope), which retrieves any arbitrary, user-defined shapes (or "features") that the user is interested in, returning solely the data within these query shapes. We show the benefits of using the Polytope algorithm as opposed to traditional data extraction methods that rely on axis-aligned coverages through specific EO examples that might be of interest to the community.

Keywords: Earth Observation, Datacubes, Feature Extraction, Data Processing, Data Management

Acronyms

EDR - Environmental Data Retrieval
EO - Earth Observation
EU - European Union
ESA - European Space Agency
I/O - Input/Output
OGC - Open Geospatial Consortium
SAR - Synthetic Aperture Radar

1. Introduction

Since the 1950s with the Explorer satellite program, which first detected the van Allen belts and studied Earth's atmospheric layers, Earth Observation (EO) has become one of the key drivers of investment in the space sector. With over a fifth of all orbiting satellites being EO satellites, this field is indeed central to the aerospace industry and is only bound to continue growing in the coming years.

EO satellites' main purpose is to collect complete datasets of various instrument measurements of our Earth and its atmosphere. These datasets are gigantic, with the European Union (EU)'s Sentinel satellites alone producing over 20 terabytes of data per day [1]. What drives the interest in EO is the ability to use the datasets produced in downstream applications and infer from them useful information such as potential extreme events for example.

The size of these datasets is however quickly growing. Indeed, with each new EO satellite launched into space and additionally, with improved on-board instrument precision, more and more EO data is produced daily. As these EO datasets grow, accessing this data becomes increasingly difficult. This poses an essential challenge to scientists trying to use this

data in downstream applications. To continue extracting value from EO satellites, we thus need to develop novel structures and methods to store and access these growing petabyte size datasets.
In this paper, we introduce an extraction algorithm, called Polytope, which helps users gain access to such peta-byte scale EO datasets. We first start by discussing how we can restructure EO data into datacubes. In particular, we highlight current initiatives that work towards this aim, before explaining how we could construct an ideal universal EO datacube. We then introduce our novel data extraction algorithm, Polytope, which provides easy and efficient access to large datacubes. Finally, we give a few examples of how Polytope can be used on EO data before concluding our work.

2. Data Challenges

With the surge of EO data produced daily in recent years, working with this data has become increasingly difficult. Whilst increased on-board instrument precision and better earth coverage hold a promise for better scientific predictions for downstream applications, efficiently storing petabytes of observation data and making it easily accessible in scientific applications has proven to be a non-trivial task.
As highlighted in [2], currently users have to download whole datasets instead of specific subsets of interest when working with EO data. Due to their size, downloading these EO datasets takes up significant disk space. When working on several of these datasets at once, users thus quickly face memory problems. This implies that users are restricted with respect to the number of datasets they can work on at the same time. As many scientific applications require the simultaneous use of different datasets, this memory limitation poses a threat to innovative downstream applications.

3. Storing EO Data

The first step towards making EO data readily available to users is to structure and store this data using a common, logical and well-defined format.
Indeed, EO data would be much easier to access if it were all stored in the same place and users did not have to browse through different platforms to obtain all of the data they are interested in. Before aggregating all EO data together however, clear guidelines on how to structure this common dataset need to be formulated. Such guidelines will help users navigate the dataset and thus make the data much more discoverable and easily accessible than before.

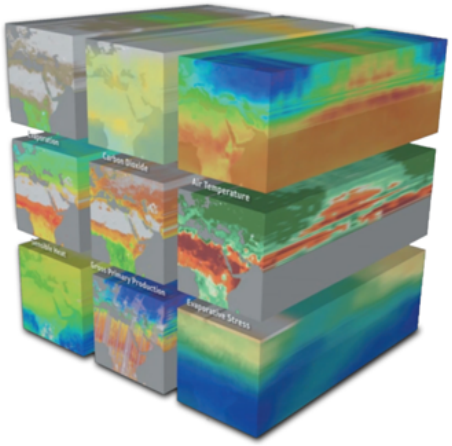


Fig. 1. Earth System Data Cube by the European Space Agency (ESA) [3].

3.1 Datacubes

The current approach taken towards efficiently storing EO datasets is to represent the data as a datacube. This is shown in Figure 1 for an example EO datacube.
The main advantage that datacubes offer compared to other data structures is that all the data is stored in a clear fashion along previously well-defined dimensions, such as latitude, longitude and time for EO datacubes. This makes browsing through petabytes of EO data extremely easy for users. Indeed, storing data this way allows users to query all different types of EO data in the same way through a common interface. In particular, knowing the datacube dimensions in advance implies that users can systematically request data along dimensions they know exist in the datacube.
As we will see in the next section, storing data in datacubes also makes it possible to take a unified approach towards efficiently extracting EO data.

3.2 Current Initiatives

After realising the importance of making EO data easily accessible to users, several entities have already started implementing their own separate datacubes. From national governments to research institutes, scientists around the world have been investigating how to group different data measurements together into datacubes. These initiatives include the Earth System Data Cube [4], developed as part of the European Space Agency (ESA) funded Earth System Data Lab [3], the Swiss Data Cube [5], Digital Earth Australia's Open Data Cube [6] (which was initially based on the Australian Geoscience Data Cube [7]), the Euro Data Cube [8] and University of Wurzburg's eo2cube [9]. We now discuss some of these datacubes and their features in more detail.

Earth System Data Cube This 4-dimensional datacube stores satellite imagery data along the spatial dimensions x and y, the time dimension t and an additional dimension *variable*, which indicates the observation type. To obtain a common grid for all observation types, regridding and gap-filling of observations is performed. Note that during these data manipulations, all original observations are kept so that no real data is lost. Where synthetic interpolated data is created, it is specifically tagged to be easily differentiated from real observations. The datasets stored in this datacube cover the entire Earth. Regions in the spatial dimensions are moreover clearly separated into land and sea regions to make different observation types available on these separate regions. This is an important feature of the datacube, which makes it possible to include observations which might only exist for exactly one of these regions, such as sea ice thickness for example. Data can then be extracted from this datacube through orthogonal subsetting operations, such as choosing specific values or value ranges to extract.

Open Data Cube This datacube stores about 1.5 petabytes of satellite imagery data along the same 4 dimensions x, y, t and *variable* as the Earth System Data Cube. All observation types in this datacube have been modified to have common metadata standards and thus make this data easier to find within the datacube. Note however that all of the original data points have been preserved here so that no information is lost. In addition, all data in this datacube has been pre-processed to make it directly analysis-ready upon retrieval. Importantly, observe that in this datacube, only local data over Australia is stored. The data here is stored using the XArray Python library [10]. To query data on this datacube, users can then use the Open Geospatial Consortium (OGC) Environmental Data Retrieval (EDR) standard [11], requesting specific pre-defined shapes, such as points, lines and polygons along the spatial dimensions. Internally, when users request a non-orthogonal shape such as a polygon which cannot be obtained by simple orthogonal subsetting, the minimum and maximum bounds of each dimension in this shape are computed. All the data which falls within these bounds is then returned to the user, so that users only ever get back bounding boxes of data.

eo2cube This project is comprised of several local 4-dimensional datacubes, storing data over local regions such as Bavaria or South Africa. The dimensions here are again the same as the ones used in the Earth System Data Cube. Compared to the other datacubes, the data here is not regridded, but instead, the original datasets are stacked together to obtain a datacube. Like the Open Data Cube, the eo2cubes are stored as XArray datasets from which users can request points, lines and polygons. Internally, when users request a shape, all the data within the minimum and maximum bounds of each dimension in the shape is loaded into memory. A mask is then applied to this extracted data to only show values over the requested shape. Some statistical computations, such as calculating the mean or standard deviation of a variable over the requested shape for example, can be automatically performed after extraction on these datacubes. This post-processing step allows users to get a brief summary and a better understanding of the data that they are extracting.

4. Polytope Extraction Algorithm

Now that we have addressed the first challenge of designing appropriate EO datacubes, we tackle the problem of extracting data from these datacubes. In particular, we introduce our new data extraction algorithm, Polytope, and give a motivation of why this

new algorithm is needed for efficiently working with EO data.

4.1 Motivation

As mentioned in [12], one of the big challenges that comes with working with EO data is downloading the data in the first place before being able to use it in downstream applications. Indeed, with a growing EO satellite population as well as increasing on-board instrument resolutions, EO data will soon reach the petabyte mark in daily data production. Users then have to select and download the datasets and specific images they are interested in among this data for different timesteps and spatial locations. This is a very expensive task both in terms of time and computer memory.

The current approach taken to perform this task is to store the data as a datacube and then query specific values or ranges across the datacube's dimensions. This orthogonal subsetting operation across the datacube dimensions was clearly highlighted in the cube examples from the previous section and can be viewed as extracting a bounding box of the data users are interested in. Most current implementations of the OGC EDR standard [11] take a similar approach and do not load in memory only the data requested, but rather the entire bounding box around it. This implies that a lot more data than users actually need is downloaded and read from the datacube's I/O system. This can lead to high latency, and in some cases, the inability to even load the requested data.

As the size of the data that users are requesting is quickly increasing, it is thus necessary to find an alternative approach to retrieve data from this EO datacube.

4.2 Polytope Mechanism

To try to address some of the issues raised in the previous subsection and be able to efficiently query data from datacubes, we have developed the Polytope extraction algorithm. This algorithm works on structured datacubes which can also be irregular and imbalanced, like the EO datacubes we described above. The idea behind the Polytope algorithm is quite simple. Users start by first providing a polytope, or n-dimensional polygon, as an input to the algorithm. As polytopes can be used to approximate any more generic shape, even shapes with smooth curved surfaces, this implies that users can in fact request any imaginable shape they are interested in. The Polytope algorithm then acts as a function which takes in this polytope shape and only returns exactly the bytes of data contained within that requested polytope in the datacube. This is shown in Figure 2 for an example of temperature data extraction over Greece.

Internally, the Polytope algorithm works by successively hyper-plane slicing the provided n-dimensional polytopes down to a list of 0-dimensional points contained within the shapes. This implies that the algorithm only finds the exact points in the shape and no additional ones. This is then used to find only the exact bytes in the data corresponding to those points.

By finding the specific bytes corresponding to the data that users want in the datacube, the Polytope algorithm makes it possible to return only these specific bytes needed by the users instead of the entire bounding boxes around the data that they want. As there are many cases where users want to extract shapes that are not bounding boxes, the Polytope algorithm thus represents a significant improvement compared to traditional data extraction approaches.

5. Example Polytope Use Cases

We now show specific examples of how the Polytope algorithm can be used in EO use cases. We also highlight the benefits of using the Polytope algorithm instead of the more traditional extraction approaches implemented in today's EO datacubes.

5.1 Sea Ice Thickness Example

Imagine a user who wants to estimate the effects of climate change on the sea ice thickness along the coastline of the Svalbard Island in Norway. They are specifically interested in comparing the difference in sea ice thickness between 1967 and 2020 in this region. Using a traditional extraction approach, they would first have to download the entire dataset of sea ice thickness around the world at their chosen timesteps in 1967 and 2020. They would then have to manually select their region of interest, but only

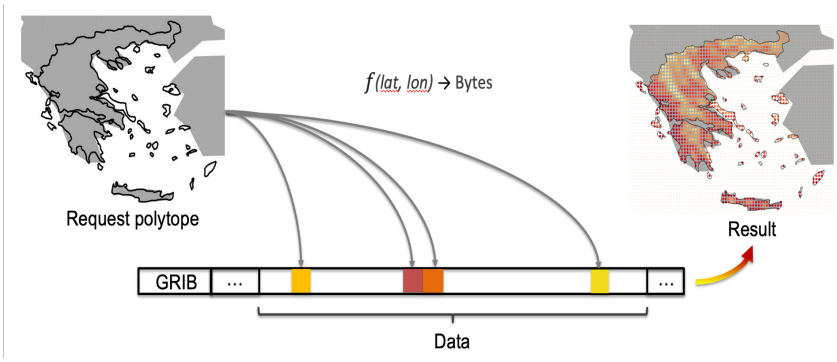


Fig. 2. Mechanism of the Polytope algorithm for an extraction of temperature data stored in a GRIB file over Greece. The Polytope algorithm acts as a function f that finds the specific bytes of temperature data contained within Greece in a given datacube.

after having downloaded sea ice thickness data for all locations over the Earth. By using Polytope, this process is made easier and the user only downloads the specific bytes of data which are of interest to them. This saves them both considerable post-processing efforts but also memory space on their local machine to which they download the data. The precise bytes the user gets back are shown in Figure 3, where the area contained within the black border line is the sea ice region around the Svalbard Island in 1967 and the blue dots represent the sea ice region in 2020, which has considerably shrunken.

5.2 Burnt Land Example

Now imagine a second user who wants to evaluate the damages made by a recent fire around his hometown in Arizona using satellite imagery. Traditionally, he would have had to download all of the satellite imagery data for a large region of around 400 km^2 around his hometown from the Landsat satellite database [15], which are about 600 KB each. Using the Polytope algorithm, he can now instead only download the few data points of interest to him around his hometown, as shown in Figure 4. From this data, he quickly determines that only about half of the land in his hometown was severely affected by the fire and burned (in red), whereas the other half was not affected at all (in yellow). The Polytope algorithm thus helps him more efficiently access the data that he needs while saving memory on his internal system.

5.3 Sea Chlorophyll Example

Finally, imagine a user who wants to assess whether a particular shipping route from Paris to New York might harm whales in the Atlantic. She is interested in the chlorophyll levels along that shipping route in the Atlantic, as chlorophyll levels are proportional to the plankton population size, which are whales' main food source. Using a traditional extraction method, she would have to extract the whole chlorophyll satellite imagery data from around the globe to then only use a small subset of this data along her shipping route of interest. Using the Polytope algorithm, she can instead directly download the data of interest to her, as shown in Figure 5. As in the other examples, this saves her significant local memory space as well as post-processing time after extraction.

5.4 Discussion

As we saw in the examples above, the Polytope algorithm can be used in many EO use cases. Compared to traditional datacube extraction methods, such as the ones used in the Open Data Cube or eo2cube, which return the entire satellite imagery datasets which contain the shape the user requested, the Polytope algorithm only returns precisely the bytes of in-

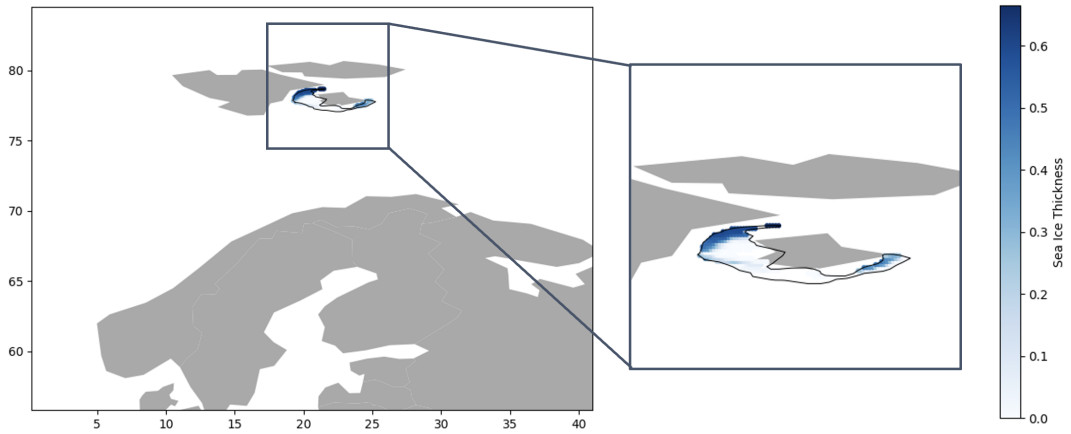


Fig. 3. Example of sea ice thickness data extraction around the Svalbard region using the Polytope algorithm. This figure was obtained by extracting the sea ice region in 1967 (region contained within the black line) [13] from the Copernicus Marine Data Store's Global Ocean Physics Reanalysis dataset [14]. The blue points are the bytes found by the Polytope algorithm for the sea ice thickness (in meters) in 2020.

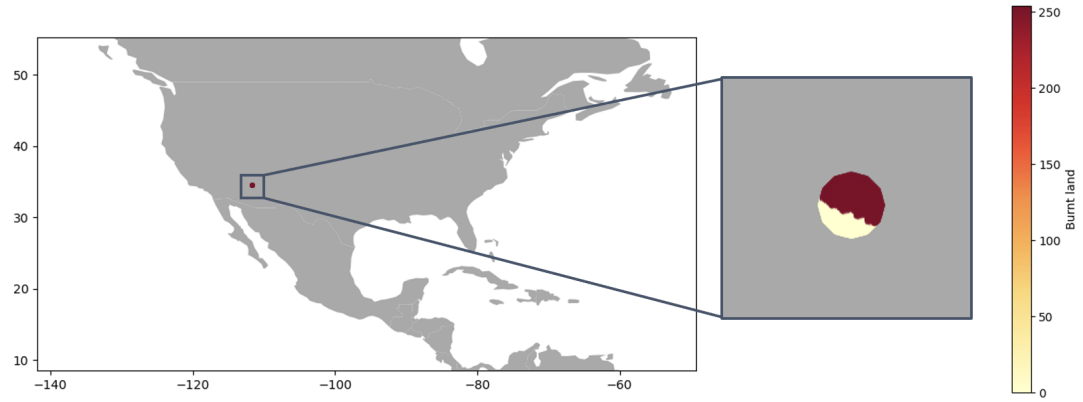


Fig. 4. Example of burnt land data extraction around a town in Arizona using the Polytope algorithm. This figure was obtained by extracting a disk around the town of interest from the USGS burnt land product (which is derived from Landsat satellite imagery data) [15]. The red and yellow points respectively represent burnt and unburnt land.

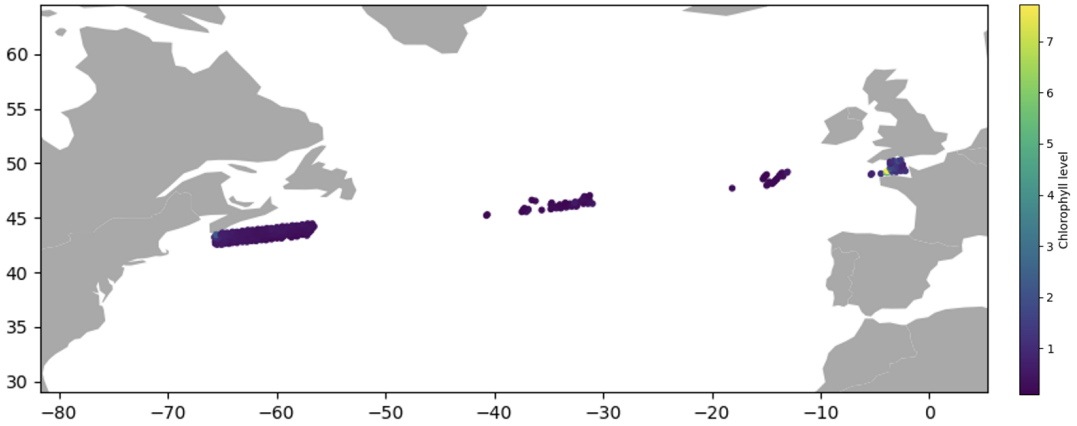


Fig. 5. Example of the chlorophyll level data extraction along a shipping route in the Atlantic using the Polytope algorithm. This figure was obtained by extracting a path shape from the Copernicus Marine Data Store's Global Ocean Biogeochemistry Analysis and Forecast dataset [16]. The coloured points are the data points found along the requested shipping route in this dataset and show the chlorophyll levels at each found location.

terest to the user. For users, this implies less post-processing and more importantly, less data downloaded locally. Furthermore, for the datacube I/O system, this lessens the load per user request on the system, potentially allowing for more users to efficiently request data at the same time. We thus see that the Polytope algorithm improves upon traditional data extraction methods both in terms of efficiency of the datacube as well as in convenience for EO data users. As we move towards larger EO datacubes, tools such as the Polytope algorithm will become essential to efficiently access EO data.

6. Conclusion

In this paper, we first discussed the need for datacubes in the EO field, describing in particular the need for a common universal datacube to make data more easily accessible to users. We then introduced our novel Polytope extraction algorithm and highlighted its benefits compared to traditional data extraction methods with the help of a few examples. Future work includes a performance analysis of the Polytope algorithm, as well as prototyping the Polytope algorithm in a EO datacube. Now that the Polytope algorithm has been shown to work in a few examples, it is essential to carry out this work before being able to use this novel extraction method in operational frameworks.

7. Aknowledgments

The work presented in this paper has been produced in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] European Commission, "New interface makes

open Earth Observation data truly open." , https://ec.europa.eu/research-and-innovation/en/projects/success-stories/all/new-interface-makes-open-earth-observation-data-truly-open, 2021. Accessed on May 24 2023.

[2] Copernicus Programme, "OBSERVER: Data cubes: Enabling and facilitating Earth Observation applications." , https://www.copernicus.eu/en/news/news/observer-data-cubes-enabling-and-facilitating-earth-observation-applications, 2021. Accessed on May 24 2023.

[3] European Space Agency, "Earth System Data Lab." , https://www.earthsystemdatalab.net. Accessed on May 28 2023.

[4] Earth System Data Lab, "Earth System Data Cube." , https://cablab.readthedocs.io/en/latest/. Accessed on May 28 2023.

[5] B. Chatenoux, J.-P. Richard, D. Small, C. Roeoesli, V. Wingate, C. Poussin, D. Rodila, P. Peduzzi, C. Steinmeier, C. Ginzler, *et al.*, "The swiss data cube, analysis ready data archive using earth observations of switzerland," *Scientific data*, vol. 8, no. 1, p. 295, 2021.

[6] Digital Earth Australia, "Open Data Cube." , https://www.dea.ga.gov.au/about/open-data-cube. Accessed on May 28 2023.

[7] A. Lewis, S. Oliver, L. Lymburner, B. Evans, L. Wyborn, N. Mueller, G. Raevksi, J. Hooke, R. Woodcock, J. Sixsmith, *et al.*, "The australian geoscience data cube—foundations and lessons learned," *Remote Sensing of Environment*, vol. 202, pp. 276–292, 2017.

[8] Euro Data Cube, "Euro Data Cube." , https://eurodatacube.com. Accessed on May 28 2023.

[9] University of Wurzburg, "eo2cube." , https://datacube.remote-sensing.org. Accessed on May 28 2023.

[10] S. Hoyer and J. Hamman, "xarray: ND labeled arrays and datasets in Python," *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[11] M. Burgoyne, D. Blodgett, C. Heazel, and C. Little, "OGC API-Environmental Data Retrieval Standard," *Open Geospatial Consortium Inc., Wayland, MA, USA, OpenGIS® Implementation Specification OGC*.

[12] G. Giuliani, G. Camara, B. Killough, and S. Minchin, *Earth Observation Data Cubes*.

[13] The University of Texas at Austin GeoData, "Arctic Ice Chart, April 1, 1967." , https://geodata.lib.utexas.edu/catalog/stanford-wz014rh6670. Accessed on May 28 2023.

[14] Copernicus Marine Data Store, "Copernicus Marine Data Store Global Ocean Physics Reanalysis Dataset." , https://data.marine.copernicus.eu/product/GLOBAL_MULTIYEAR_PHY_001_030/description. Accessed on May 28 2023.

[15] US Geological Survey, "Usgs data." , https://earthexplorer.usgs.gov. Accessed on May 28 2023.

[16] Copernicus Marine Data Store, "Copernicus Marine Data Store Global Ocean Biogeochemistry Analysis and Forecast Dataset." , https://data.marine.copernicus.eu/product/GLOBAL_ANALYSIS_FORECAST_BIO_001_028/description. Accessed on May 28 2023.

6 Feature Extraction on Unstructured Grids

So far, we have demonstrated that the Polytope algorithm is both efficient and scalable across a wide range of datasets and application domains. One important limitation of the current approach, however, is its restriction to well-structured isolatitude grids. As some NWP models and other scientific simulations are produced on fully unstructured grids, they therefore fall outside the scope of the original algorithm.
We address this limitation in the following paper [72] by extending the Polytope algorithm introduced in Chapter 2 to support unstructured grids. This extension introduces a second feature extraction algorithm that relies on fewer structural assumptions and can therefore operate on arbitrary grid types. We present examples of applying the extended algorithm to several unstructured grid configurations and evaluate its performance and scalability, with particular emphasis on comparisons to the original algorithm. Finally, we discuss how this new approach can be integrated into the broader Polytope-based feature extraction service as part of a unified data extraction system.

This work has been submitted for publication in the 2026 proceedings of the Platform for Advanced Scientific Computing (PASC) and the open-source implementation of the algorithm is available on GitHub as part of the Polytope repository [75]. I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *J. Hawkes*: Conceptualisation, Writing-review & editing; *M. Schultz*: Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

An Algorithm for Feature Extraction from Large Scale Meteorological Data Stores on Irregular Grids

Mathilde Leuridan
European Centre for Medium-Range Weather Forecasts (ECMWF) and University of Cologne
mathilde.leuridan@ecmwf.int

James Hawkes
European Centre for Medium-Range Weather Forecasts (ECMWF)
james.hawkes@ecmwf.int

Tiago Quintino
European Centre for Medium-Range Weather Forecasts (ECMWF)
tiago.quintino@ecmwf.int

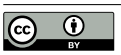
Martin Schultz
Forschungszentrum Juelich (FZJ) and University of Cologne
m.schultz@fz-juelich.de

Abstract

In recent years, the volume of meteorological data has grown rapidly due to higher model resolutions and more frequent data output. To handle this increase efficiently, the European Centre for Medium-Range Weather Forecasts has developed new ways to extract data without accessing full meteorological fields through their feature extraction capabilities. This includes in particular the Polytope algorithm, which lets users extract time series or regional subsets directly from the data backends, greatly reducing data transfer and processing time. However, the current algorithm only works on structured iso-latitude grids such as octahedral or HEALPix grids. Many modern meteorological models, by contrast, use unstructured grids where points do not align along latitudes, such as Lambert conformal or ICON icosahedral grids. In this paper, we extend the Polytope algorithm to support these irregular grids. We first explain why the original approach was limited to structured data before introducing a new extraction method, which uses the well-studied quadtree data structure. As a popular data structure in geospatial applications, we focus on describing how quadtrees can be integrated within the Polytope feature extraction framework to handle complex grid geometries. We then assess the performance and scalability of this algorithm, which performs on par with state-of-the-art alternatives used in geospatial applications. In particular, the extended algorithm outperforms other methods when accessing large or complex regions and therefore offers a compelling alternative to current data extraction techniques for large-scale datasets. Finally, we conclude by discussing how this algorithm can be integrated into an operational data service.

CCS Concepts

• **Mathematics of computing** → **Mathematical software performance**; • **Information systems** → **Data access methods**; **Extraction, transformation and loading**.



PASC '26, Bern, Switzerland

ACM ISBN 979-8-4007-1886-1/2025/06
https://doi.org/10.1145/3732775.3733573

Keywords

Data Extraction, Data Management, Numerical Weather Prediction

ACM Reference Format:
Mathilde Leuridan, James Hawkes, Tiago Quintino, and Martin Schultz. 2026. An Algorithm for Feature Extraction from Large Scale Meteorological Data Stores on Irregular Grids. In *Platform for Advanced Scientific Computing Conference (PASC '26), June 29–July 1, 2026, Bern, Switzerland.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3732775.3733573

1 Introduction

In recent years, scientific data production has reached unprecedented scales, with scientific data archives around the world growing by orders of magnitude. In the fields of numerical weather prediction (NWP) and climate science in particular, initiatives such as kilometer-scale modeling [14, 23, 29, 50] and AI-driven weather forecasting [6, 25, 44] have pushed daily data generation into the petabyte range. With these increasing data volumes, and with the rise of data-driven science and engineering, efficient data access has become essential for enabling scientific discovery, innovation and timely decision-making.

In meteorology, data is typically stored as two-dimensional horizontal slices of the atmosphere, called fields. Traditionally, meteorological data access systems return entire fields to users, without the possibility to subset these data units. However, with model resolution increases, field sizes start to grow quadratically. This fast growth can then quickly strain the data systems, especially when serving many users at once. To maintain scalability, modern data frameworks are shifting towards the use of chunking [3, 38, 43] to enable efficient data access. This approach, however, depends on prior knowledge of common access patterns to optimize chunk layout.

To support arbitrary data access without such assumptions, the European Centre for Medium Range Weather Forecasts (ECMWF) developed the Polytope feature extraction service [27, 28]. Instead of relying on pre-defined chunks, Polytope dynamically computes which data bytes to query from meteorological data files. This enables the flexible and efficient retrieval of any high-dimensional shape from large weather datacubes, supporting fast on-demand access for dynamic digital twins and helping build more streamlined, user-centric workflows. Polytope is an efficient data retrieval technique that reduces both the load on the system, as well as the

user-side post-processing needs [26]. Using this algorithm, a scientist estimating storm risk for a specific city can, for example, extract only the relevant grid points, rather than downloading full forecast fields, as required by traditional systems. This minimises data transfer and simplifies downstream analysis. Polytope has not only demonstrated strong performance across a variety of use cases, but also scales well to different extraction shapes and dimensions. However, despite its advantages, Polytope currently only supports extraction from datacubes on iso-latitude grids. Whilst this includes widely used grids like the octahedral [36], HEALPix [15], and regular latitude-longitude grids, it is a significant limitation of the current algorithm. In particular, it implies that feature extraction is not currently possible on more complex irregular grids, such as the icosahedral grid [32] used by the ICON model [60], the tripolar and curvilinear grids [30] used by the NEMO model [34] or the Lambert conformal conic projection [9, 57].

In this paper, we address this constraint by extending Polytope to support arbitrary unstructured grids. We first provide a more complete motivation of this work, before explaining the origin of the limitation in the current algorithm and reviewing related work. We then introduce the extended algorithm, demonstrate its use on various irregular grids, and evaluate its performance. Finally, we discuss integration into the existing service and consider performance and scalability implications on the overall data extraction system.

2 Background

To motivate our work, we begin this section by quickly describing the original Polytope algorithm, before then providing a few examples of irregular grids. These examples highlight a key limitation of the current Polytope algorithm, which relies on the assumption that the datacube can be decomposed into tensor products of 1-dimensional spaces. Since irregular and unstructured grids are inherently embedded into 2-dimensional space however, a different extraction approach is required for them. To address this inconsistency, we will need to adapt the current query algorithm to operate in higher dimensional spaces. We therefore conclude the section with a review of related work on 2-dimensional point search and spatial indexing problems.

2.1 Original Polytope Algorithm

The Polytope algorithm [27] is a feature extraction algorithm which, for any arbitrary n-dimensional request shape, computes the set of datacube points contained within it. Using computational geometry concepts, it first decomposes the request shape into a set of convex sub-polytopes. At this point, each sub-polytope is then successively sliced along all of the datacube coordinate axes. This reduces the sub-polytope dimension by dimension until it has been completely decomposed into a set of one-dimensional line segments, as shown in Figure 2. The datacube points contained within these segments are then identified and returned to the user as the full set of points contained in the original request shape.

The algorithm relies on the important assumptions that we can both perform these axis-aligned slicing operations efficiently, and then quickly identify the points contained within the resulting one-dimensional segments.

2.2 Irregular Grids

While many NWP models are based on iso-latitude grids, other grid types offer interesting advantages, including improved spatial uniformity and a more stable convergence of solutions. These encompass icosahedral [54] and Yin–Yang grids [22], as well as projected grids such as Lambert conformal [57], Mercator [16] and polar stereographic projections [33], which are commonly used in limited area modelling [8, 42, 46] for their regional accuracy and numerical stability. Examples of such grids, which are not iso-latitude and we therefore treat as irregular, are shown in Figure 1(a).

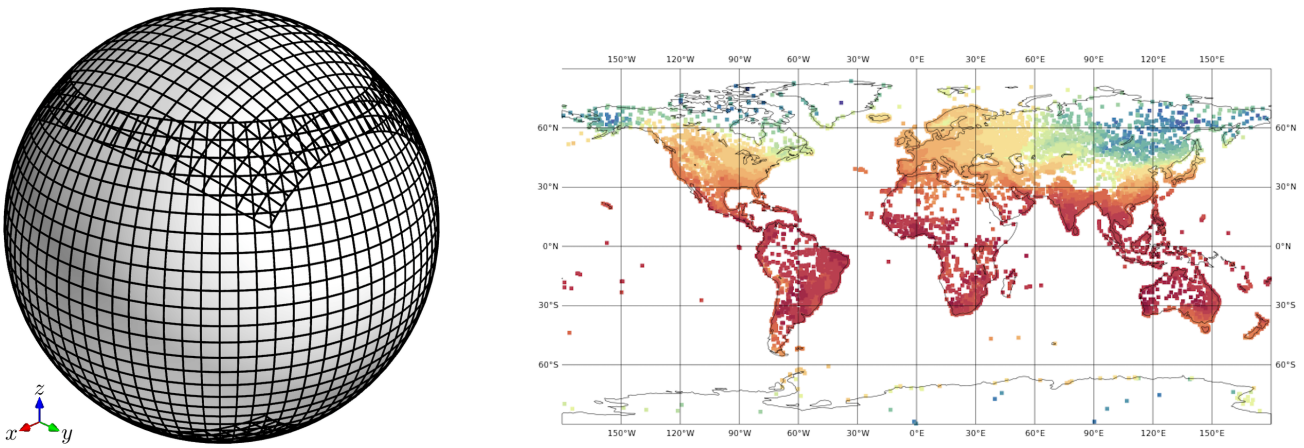
Although they are not iso-latitude, these grids still retain a clearly defined spatial arrangement. By contrast, some earth science datasets lack any regular 2-dimensional structure and are thus classified as unstructured grids [56]. Examples include observational datasets and more complex spatial configurations such as the ocean ORCA grids [35], which are especially tuned to lie within specific regions. Examples of such completely unstructured grids are shown in Figure 1(b).

In all of these examples, the points are clearly unaligned and do not follow a coherent structure along the coordinate axes. The current Polytope algorithm, however, relies on slicing polytopes along each of the datacube axes, including the grid axes, at constant coordinate values. As described in [27] and shown in Figure 2, this process eventually requires identifying points that lie within the extents of a line segment on an axis. On structured grids, this works well because multiple points can usually be found within such extents at the same time and for a relatively cheap cost. However, for unstructured and irregular grids, or more generally any non–axis-aligned point clouds, each point would need to be checked individually, since there is no guarantee that several points will fall within a given extent. This makes the operation inefficient and unsuitable in practice for unstructured data. Therefore, a different approach is needed to perform an operation comparable to the Polytope slicing in these cases.

Fundamentally, the problem arises because we try to simplify the search by reducing it to a one-dimensional space, where binary search can then be applied efficiently. This works well for uniform iso-latitude grids, since their layout can be mapped clearly onto a line. Irregular and unstructured grids however do not behave this same way as they are inherently two-dimensional objects. Forcing them into a one-dimensional order therefore loses important spatial relationships and as a result, one-dimensional methods like binary search tend to break down and become inefficient. Instead, we need to approach the problem in its natural dimensionality by using spatial indexing and search techniques that are built for higher dimensions. This includes the use of specialised data structures or other spatial partitioning methods [4, 12, 17, 52, 53], which allow for efficient point searches in irregular and unstructured spaces as they preserve the spatial aspect of the problem.

2.3 Related Work

Over the years, scientists have developed a variety of data structures and query algorithms to efficiently work with spatial point cloud data. With the rapid growth of geospatial data in particular, specialized systems and techniques have emerged to improve how this data is indexed and queried.



(a) Example of a Yin-Yang grid [2].

(b) Example of a temperature observations dataset [13].

Figure 1: Irregular grid examples.

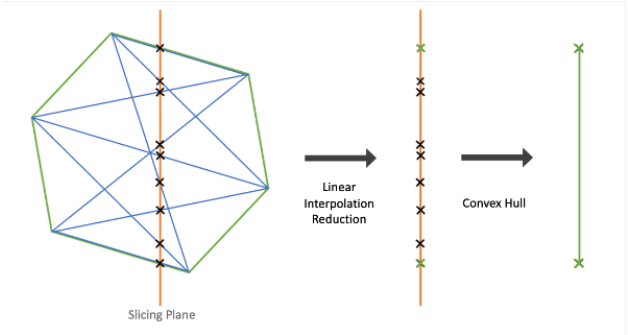


Figure 2: 1-dimensional slicing step.

One of the fundamental operations in this context is the point-in-polygon test [51], used to determine whether a point lies within a given polygon. Since its first introduction, this method has been refined many times with techniques such as ray tracing [20, 24] to increase efficiency for different use cases. In the geospatial field, the point-in-polygon test is often used in polygon range queries [40, 61]. Such queries aim to identify which points in a point cloud fall within a user-defined region of interest, and are in fact the 2-dimensional equivalent of the problem that the Polytope algorithm is trying to solve. Note, however, that the naïve approach of solving this problem by performing expensive point-in-polygon checks for every point in a point cloud becomes prohibitively expensive for high-resolution datasets with thousands or millions of samples. Instead, researchers have developed specialised spatial data structures that capture how points are distributed in space and use this additional information to avoid exhaustive and unnecessary point-in-polygon checks. Examples include k-d trees [4], R-trees [17] and quadtrees [49], along with their higher-dimensional variants [5, 31, 37]. Such spatial indexing structures have been widely adopted in geospatial databases and scientific data analysis libraries for over a decade. Notable examples include the use of k-d trees and octrees in the Point Cloud Library [47], R-trees in GeoPandas [21], k-d trees in the NextGEMS processing pipeline [50], a range of spatial tree structures within the PostGIS framework [41], and octree-based indexing in OpenVDB [39]. Each of these structures offers unique trade-offs, and we now review some of their strenghts and limitations to identify the most suitable approach for designing an alternative slicing algorithm to the current Polytope algorithm.

k-d trees. A k-d tree is a binary search tree for locating points in k-dimensional space. Each node splits the space into two half-spaces by an axis-aligned hyperplane. The splitting coordinate typically changes with depth and median splits are common for balance, though other heuristics exist. Recursion continues until leaf nodes contain a single or a small number of data points. This data structure naturally generalises to any number of dimensions, although it is most efficient in low dimensions where trees stay shallow. For very large or high-dimensional datasets, the tree can become deep as only one dimension is split per level. These trees enable fast nearest-neighbor searches and range queries by pruning subtrees whose subspace lies outside of the query. Axis-aligned box queries are especially efficient on this data structure because intersection tests with the splitting hyperplanes are simple cheap comparison operations.

R-trees. An R-tree is a hierarchical tree data structure, which stores information about how objects are clustered in k-dimensional space. In such a tree, each node stores the minimum bounding box that encloses all of its children. The tree therefore represents rectangular regions in space rather than individual points. This design makes R-trees efficient for queries on region intersection, containment and unions, particularly for rectangular or box-shaped ranges. However, exact point queries or searches with arbitrary polygons are less efficient, since they first require approximating each point with a bounding box before checking which of these boxes intersect the query region, effectively building a box coverage of the polygon. For each box of the coverage, we would then finally

need to check point containment to verify that the points in the bounding boxes are really inside of the polygon.

quadtrees. A quadtree is a spatial tree data structure used for storing two-dimensional point data. Starting from a bounding box of the domain, it is built by recursively sub-dividing the space into four equal sized quadrants. The recursion step is repeated until the resulting quadrant only contains a small number of points, when the process terminates and the points contained inside of the quadrant are stored as leaf nodes of the tree. This construction guarantees that at any given depth, all quadrants in the tree cover the same area, which ensures a uniform partitioning of the original space. Since large empty regions can easily be pruned during recursion, quadtrees are well suited for range queries, as well as for applications such as collision detection and spatial mapping. Their efficiency is however strongly correlated to the underlying data point distribution as, with uniformly spaced data, the tree remains balanced and shallow, while more clustered points lead to an unbalanced tree that requires deeper traversal to answer a query.

The main objective of the algorithm developed in this paper is to efficiently identify all of the points contained within a given polygon from a high-dimensional unstructured point cloud. The different spatial data structures described above offer complementary strengths, but not all are equally well suited to this task.
R-trees, for example, are designed to capture relationships between extended regions by storing minimum bounding boxes. Whilst this makes them effective for rectangular range queries, it complicates polygon containment problems. As already mentioned, to handle such problems in an R-tree, each point must first be represented as a bounding box, and polygon queries then require additional intersection tests. This adds unnecessary overhead because the problem is fundamentally point-based. k-d trees represent a more natural approach, as they explicitly store point locations by recursively partitioning the space along coordinate axes. However, in large and fairly homogeneous spaces, such trees become very deep, since only one dimension is split at each level. Querying then requires traversing many nodes, making this data structure computationally expensive for problems such as retrieving all points contained within a polygon. Quadtrees offer a more suitable solution in this situation. Indeed, they not only provide a balanced representation of the space, especially when points are uniformly distributed, but also enable efficient pruning of uninteresting regions while still keeping tree depth manageable. This makes polygon containment queries, such as the one we are trying to perform, both practical and efficient.
Following this discussion, we therefore choose quadtrees as the foundation of our work and design an algorithm that leverages their properties to solve the required polygon containment query.

3 Extended Slicing Algorithm

In this section, we introduce a higher-dimensional slicing algorithm that operates directly in higher dimensional spaces. This is in contrast to the original Polytope slicing mechanism, where the problem was always first reduced to lower dimensions. The method described here builds on established spatial data structures to enable efficient queries, with quadtrees chosen as the most suitable for polygon range searches. As discussed previously, such spatial data structures are well-studied and have already been applied to a range of geospatial problems. In this work however, we focus on their adaptation and integration within the Polytope framework, showing how they can be applied to large-scale meteorological data extraction from complex grid geometries in an operational setting. We first present the algorithm, before illustrating its extraction capabilities. We then finally conclude the section by evaluating the performance and scalability of the algorithm on a few example Numerical Weather Prediction (NWP) datasets.

3.1 Modified Algorithm

Our proposed algorithm extends the Polytope algorithm [27] to higher dimensions. Whilst the overall framework, including tree construction, remains the same, the slicing step requires fundamental changes. The core assumption of the original method, which is that points within a query range can be located directly as we reduce the problem to 1-dimensional spaces, breaks down in a higher-dimensional setting. This limitation makes the original algorithm inefficient and impractical, if not completely unusable. To overcome this, we instead introduce a new algorithm, which uses a quadtree structure to efficiently identify points contained within polygons. This not only addresses the weaknesses of the original approach, but also preserves scalability of the algorithm. Although the approach naturally generalises to higher dimensions, in the following we focus on two-dimensional unstructured latitude-longitude subspaces since vertical levels, as well as other dimensions such as time or ensemble number, are always well aligned in NWP and climate models.

3.1.1 Pre-Computation Step. As a first step, we introduce a pre-computation phase in which the available data in the datacube is organised into a quadtree with leaf quadrants that each contain a single grid point. This pre-processing is necessary because, unlike earlier methods that could efficiently query raw data directly, our algorithm requires a static spatial index to remain computationally cheap. Indeed, building the quadtree in advance encodes the spatial distribution of the data into a hierarchical structure, which can then be reused efficiently during polygon and range queries. By shifting this cost to pre-processing, we ensure that subsequent queries on the same dataset are faster and, crucially, also remain scalable when dealing with larger data volumes.
This pre-computation phase is especially effective for static grids, such as those on which NWP models are defined, where the quadtree only needs to be computed once and can then be reused across many queries. More dynamic grids, such as the ones defining observational datasets from mobile platforms like ships or aircraft, might change between queries and thus require the quadtree to be recomputed more frequently. Whilst this reduces the benefits of caching the quadtree, the algorithm described here still applies in such cases, with performance depending on how often the underlying grid changes.

3.1.2 Dynamic Step. Building on the precomputed quadtree, we introduce a slicing algorithm to identify all of the points contained within a user-defined polygon. The general idea is to recursively

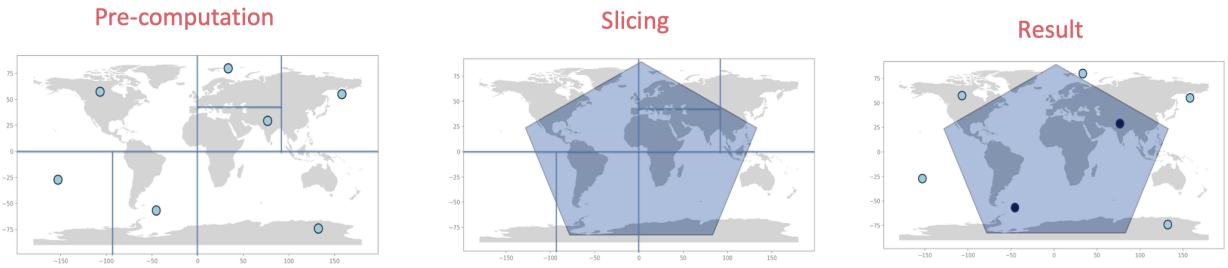


Figure 3: Unstructured grid slicing algorithm.

slice the polygon along the boundary of each of the quadtree quadrants. At each step, we act on a given quadtree node, where a vertical and a horizontal slice divide the polygon into four sub-polygons, each fully contained in one of the child quadrants of the current node. This subdivision continues until the leaves of the tree are reached, where a point-in-polygon test then confirms whether the leaf point lies inside of the final sub-polygons. The full procedure is formalised in Algorithm 1 and illustrated in Figure 3.

Our approach extends two core components of the original Polytope slicing algorithm [27]. First, the point inclusion step is generalised from one-dimensional bound checks to a full two-dimensional point-in-polygon test. Similarly to the original Polytope algorithm however, the polygon that is being sliced at this step is guaranteed to be convex. We can therefore apply similar strategies as in the original algorithm to reduce the problem to a one-dimensional line segment. As illustrated in Figure 4, the point-in-polygon test can be reformulated as two simple bound checks. First, we verify whether the vertical coordinate of the point lies within the vertical extent of the polygon and, if so, we then determine the corresponding horizontal extent of the polygon at that vertical position and check whether the point's horizontal coordinate falls within this interval. Second, the one-dimensional hyperplane slicing operation is extended to two dimensions, as shown in Algorithm 2. Instead of reducing the dimensionality of the polygon by projecting it to a lower-dimensional space, we preserve the full dimensionality of the problem here by slicing the polygon along each dimension at every recursion step. This operation produces four 2-dimensional sub-polygons, which replaces the original algorithm's 1-dimensional projection and thereby ensures that the multi-dimensional nature of the problem is maintained. Note that we perform similar steps to the original 1-dimensional algorithm here, and in particular, we still compare pairs of point on either side of the slice plane before taking the convex hull. We therefore maintain the same complexity as in the original algorithm [26] of $O(m^2)$, where m is the number of nodes defining the input polygon. These two extensions allow the algorithm to take full advantage of the hierarchical structure of the quadtree, thus making polygonal queries in two-dimensional space both efficient and scalable.

3.1.3 Optimisation Step. As an optimisation step to improve performance, the algorithm checks whether a sub-polygon exactly overlaps the full area of a quadtree node. If this is the case, the polygon fully covers the associated node quadrant, which implies that all points stored in the leaf nodes of that branch must lie inside of the polygon. These points can therefore directly be added to the search result without any further tree traversal or point-in-polygon tests. This pruning step allows an early return, avoiding unnecessary recursion, as well as helps reduce significant computational overhead, especially for larger polygon queries.

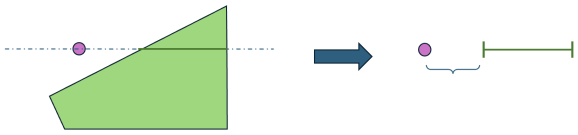


Figure 4: Custom point-in-polygon test.

3.2 Examples

We now demonstrate example extractions with our algorithm on two non-iso-latitude grids, namely a Lambert conformal projection grid and an ICON icosahedral grid. Both grids are used in the Destination Earth On-Demand Extremes digital twin to run specialised local meteorological models. Unlike regular latitude–longitude grids, these exhibit more complex two-dimensional mappings, as shown in Figure 5, where the grid points are clearly not aligned with the coordinate axes.

The Lambert Conformal projection is constructed by projecting a regular iso-latitude grid onto a cone and flattening it into the plane [9]. In principle, one could design a mapping to transform polygons into this space, apply the original Polytope algorithm in a decomposed 1D form, and then map the results back. However, this would require additional transformations and is not straightforward in practice. Our quadtree-based method sidesteps this complexity, as illustrated in Figure 5(a), which shows extractions directly on the projected grid.

The ICON grid is even less structured. It is built from a nested icosahedral decomposition of the sphere, whose vertices are by construction never aligned with latitude or longitude [32]. Unlike the Lambert projection, there is no simple analytical mapping to transform this grid into an iso-latitude representation. In practice,

Algorithm 1 Polytope Intersection Algorithm

Input: quadtree node n, polygon $\mathcal{P}$.

if n is a leaf **then**
  if n's quadrant contains a data point **then**
    if data point is contained in $\mathcal{P}$ **then**
      Add data point to results
    end if
  end if
else
  Slice $\mathcal{P}$ into polygons $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$ that fit inside n's children's quadrants
  for child node c_i in n's four children **do**
    Repeat algorithm with new inputs c_i and $\mathcal{P}_i$
  end for
end if

Algorithm 2 Slice Algorithm

Input: polygon $\mathcal{P}$, slice value x_f.

Separate $\mathcal{P}$'s vertices into sets S_l and S_r of points to the left and right of x_f.
for pairs $(p_l, p_r) \in S_l \times S_r$ **do**
  Find intersection point p_m of the line through p_l and p_r with the line $x = x_f$.
  Add p_m to a set S_m.
end for
Take convex hull of the sets $S_l \cup S_m$ and $S_r \cup S_m$ to get left and right sliced polygons.

ICON grids are distributed as precomputed files that store the point locations explicitly. This irregularity makes polygon slicing particularly challenging. Nevertheless, as Figure 5(b) shows, our algorithm successfully extracts subsets of points on this unstructured grid without requiring such a mapping.

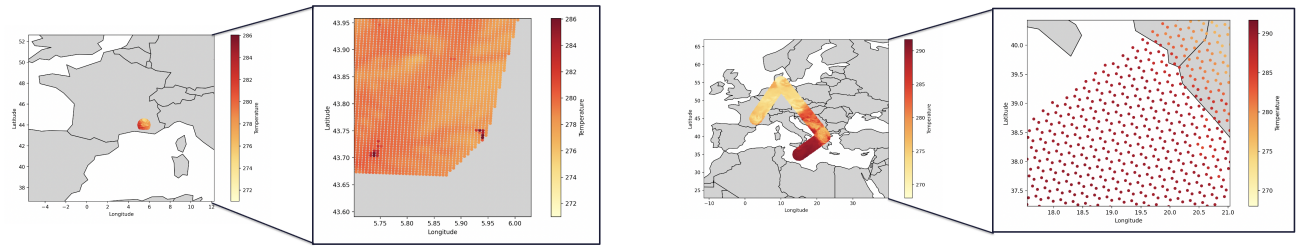
These examples highlight that the algorithm works reliably not only on regular grids but also on complex, non-axis-aligned structures. This allows us to return only the small fraction of bytes relevant to a user's query, even for very large, high-resolution datasets, preserving the original motivation behind the Polytope algorithm.

3.3 Performance

The performance of the extended algorithm depends on two important aspects, which are the complexity of the input shape and the complexity of the underlying dataset. Dataset complexity is influenced by grid resolution and point distribution, which both determine the number and density of points to be processed. Shape complexity is defined by factors such as its area, the number of vertices and the number of unions defining the shape. To assess how these factors affect performance in an operational setting, we first analyse their individual scaling behaviour. We then compare the algorithm against state-of-the-art point-in-polygon approaches, which serve a similar purpose. These reference methods rely on standard point-in-polygon tests, often optimised by limiting computations to points within a polygon's bounding box, which is an approach that we also adopt for our comparison [55, 58, 59]. The algorithm implementation used for this performance analysis is written in Rust and does not make use of any parallelism.

3.3.1 Scaling. The scaling behaviour of the algorithm is influenced by two main groups of factors, those related to the size and structure of the dataset and those linked to the complexity of the input polygon. While shape-related factors primarily affect the dynamic extraction time, dataset-related factors impact both the quadtree construction, or pre-computation phase, as well as the dynamic extraction time.

We first investigate how the characteristics of the underlying dataset affect total extraction times. In this experiment, a box of constant size is extracted from uniform point clouds of varying densities to assess how Algorithm 1 scales. As shown in Figure 6(a), both the quadtree construction time and the polygon point extraction time scale linearly with the number of points in such a uniformly distributed point cloud. This behaviour is expected as a homogeneous point distribution produces a well-balanced quadtree, whose traversal depth increases proportionally with the number of points. However, real-world datasets rarely exhibit such uniformity. To assess the effect of spatial inhomogeneity, we generate point clouds of one million points drawn from normal distributions with varying standard deviations around the centre. Figure 6(b) shows that when points are clustered and less evenly distributed, the quadtree becomes less balanced, leading to higher construction times. On the other hand, when points are more dispersed in space, the time required to identify points within the input shape increases. Although this behaviour may seem counterintuitive, it arises because a more dispersed distribution reduces opportunities for early termination in the traversal. Indeed, since fewer quadrants are fully contained within the selection area, more of the quadtree must be explored during the algorithm runtime.



(a) Extraction on a local Lambert conformal projection grid.

(b) Extraction on an ICON grid.

Figure 5: Extraction examples on irregular grids.

Besides the point cloud properties, the remaining key factor which influences the algorithm's performance is the complexity of the input polygon. Polygon complexity is mainly determined by two characteristics, the number of vertices defining its boundary and the number of polygon unions which compose the overall shape.
We first examine the impact of the vertex count. To isolate this effect, we use polygonal approximations of a circle with progressively more vertices, keeping the enclosed area approximately constant to avoid bias. The results, shown in Figure 7(a), confirm that quadtree construction time remains constant, as the underlying dataset is unchanged. For polygons with few vertices, total extraction time is initially dominated by data retrieval from the storage backend via the GribJump software [11], while the computational cost of determining points within the polygon is low. This occurs because simpler polygons cover larger areas, leading to larger data extractions. As polygon complexity increases, backend data retrieval time stabilises, which indicates a consistent number of retrieved points, whilst the time to compute points contained in the polygon grows quadratically. This behaviour aligns with the algorithm's theoretical scaling, as the slicing step involves an increasing number of polygon–line intersection operations.
The second factor to investigate is the number of unions composing the shape. To study this, we repeat the extraction for the same circular region, represented as a union of an increasing number of sub-polygons. As shown in Figure 7(b), both quadtree construction and data retrieval times remain constant, since the underlying point cloud and extracted area are unchanged. However, the time required to compute points within the polygon grows linearly with the number of unions, which then directly impacts the total extraction time. This linear relationship can be explained by the fact that each sub-polygon is processed independently using Algorithm 1.

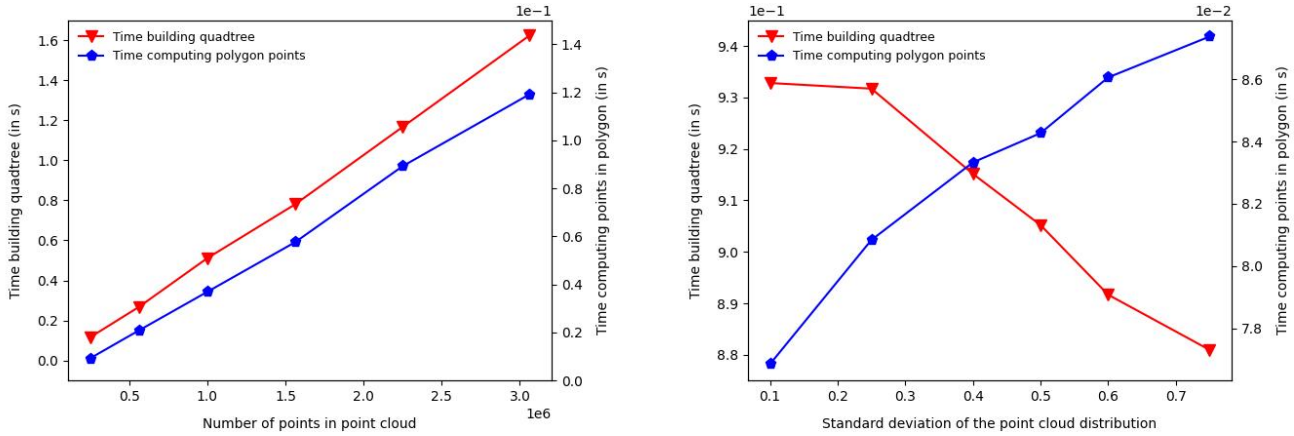
3.3.2 Method Comparison. Having examined the key factors influencing extraction performance from both the dataset and polygon perspectives, we now compare the algorithm against state-of-the-art, highly optimised point-in-polygon (PIP) methods [18, 19, 61]. Specifically, we benchmark against the Rust geo library implementation [45], which stores the input polygon in memory and employs bounding-box skipping to avoid unnecessary point checks. To evaluate performance, we perform two sets of experiments, one using progressively larger rectangular boxes and another using triangles of increasing size but contained within the same bounding box. In both cases, the polygons are simple, allowing the baseline PIP tests to perform efficiently.
Figure 8(a) presents the results for the box experiments. The quadtree-based approach performs comparably to, and often slightly better than, the optimised PIP implementation, particularly for larger boxes containing more points, where the number of required PIP checks increases substantially. Similar trends are observed for the triangle experiments in Figure 8(b), where the quadtree method outperforms the PIP tests again. In particular, we note that the performance gap widens for larger extraction areas. Both of these results also confirm that the quadtree algorithm scales linearly with the area of the extracted region.
It is finally worth noting that standard PIP tests degrade in performance as polygon complexity increases, due to the rising cost of ray-casting operations. In contrast, the quadtree approach stays more efficient while offering additional functionality, such as nearest-neighbour searches [1, 48], at minimal additional computational costs. We therefore conclude that our quadtree-based algorithm achieves performance on par with leading PIP methods used in GIS and Earth science data analysis, while providing enhanced versatility for meteorological feature extraction across diverse grid structures.

4 Discussion

As shown throughout this paper, the new slicing algorithm enables point searches within arbitrary polygons on fully unstructured spaces, without assumptions on the spatial arrangement of points. This represents a major improvement over the original Polytope algorithm and, in particular, makes it possible to serve datasets that were previously unsupported. Since many meteorological services use diverse and non-standard grids, the relaxed constraints of our approach extend the Polytope service to virtually all dataset types. However, deploying this algorithm in real-world scenarios, especially as part of an operational service, requires addressing several practical considerations, which we now discuss.

4.1 Data Extraction as a Service

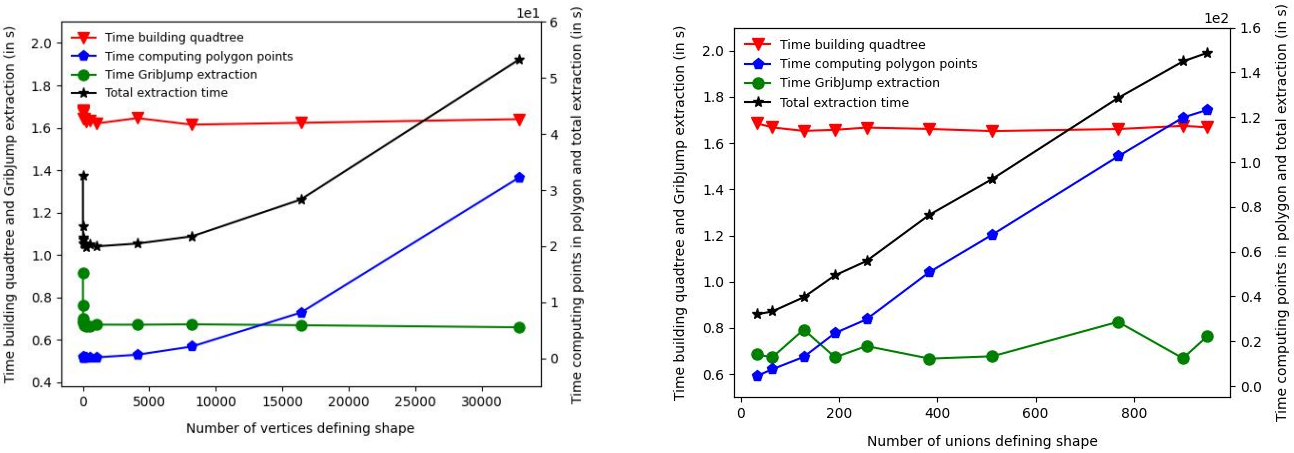
To make the described slicing capability accessible in practice, the algorithm must be integrated into an operational service that allows users to extract subsets of data directly from large multidimensional



(a) Varying homogeneous point density.

(b) Varying inhomogeneous point distribution.

Figure 6: Extraction times of a box for varying point clouds.



(a) Varying number of vertices.

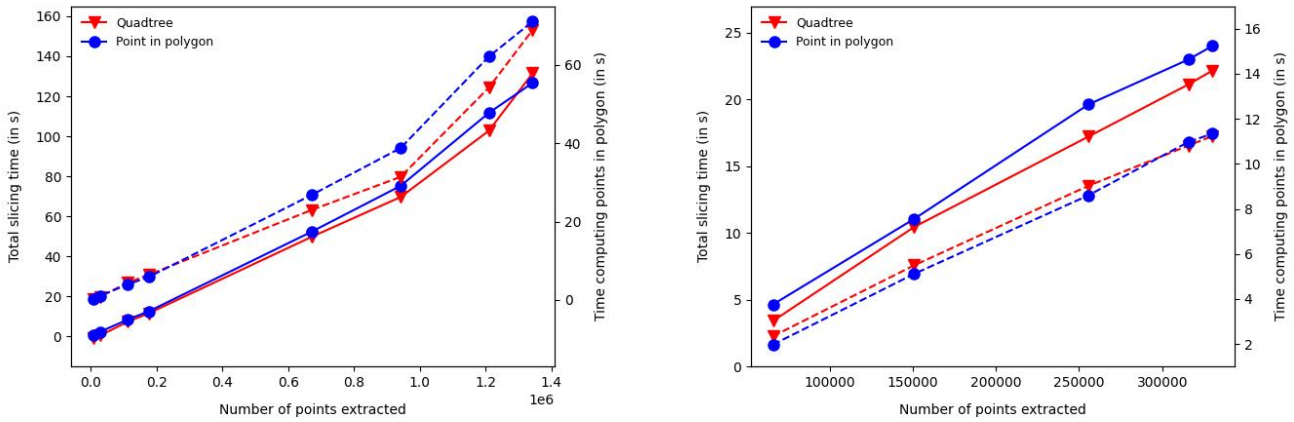
(b) Varying number of sub-polygon unions.

Figure 7: Extraction times for a disk defined with a varying number of vertices (a) and a varying number of sub-polygon unions (b).

datasets. Such a service should efficiently handle different grid structures while maintaining responsiveness for on-demand queries.

Whilst the new slicing algorithm greatly extends applicability to unstructured grids, it also introduces significant additional computational costs due to the preprocessing required to construct the quadtree. In contrast, the original Polytope slicer remains more efficient for structured, iso-latitude grids where such preprocessing is unnecessary. On top of these few seconds of additional preprocessing, the original Polytope slicer will moreover benefit from knowledge of the data structure during slicing and typically achieves a quadratic speedup over the fully unstructured version. To maintain both flexibility and high performance, a complete data extraction service must therefore integrate both algorithms in a single mechanism by dynamically selecting the most suitable slicer based on the dataset and its grid type.

Beyond considerations of computational efficiency, using this new algorithm as part of an operational service introduces additional memory management challenges. Since the method depends on precomputed quadtrees, these data structures must be stored persistently for static reuse or regenerated dynamically for each incoming request. Depending on system architecture, this can be adressed by caching quadtrees for commonly accessed grids, distributing them



(a) Increasing square boxes.

(b) Skewed triangles contained in same bounding box.

Figure 8: Comparison of extraction times between quadtree approach (Algorithm 1) and optimised point in polygon test for different shapes, where the solid lines show the time computing the points in the polygon and the dashed lines show the total slicing time.

across sub-services or applying compression when decompression costs are acceptable.

Considering these aspects is crucial to ensure that an operational service can efficiently deliver data to users, minimising computational overhead while preserving scalability and versatility across supported grid types.

4.2 Future Work

The service described above works reliably as long as the grid type and configuration of the dataset on which we want to apply the algorithm are known in advance. A key direction for future work is to automate this step through a data discovery mechanism capable of identifying the underlying grid of any dataset. This could for example be achieved by introducing a new data structure that stores metadata about each dataset, including its grid specification, as well as other relevant spatial characteristics. Such an approach would enable a fully dynamic service that automatically selects the appropriate slicer without requiring prior user input. Beyond improving user-friendliness, this would more importantly enable streamlined and interactive access to data, paving the way for integration of the service within a digital twin ecosystem. As the ultimate objective is to power digital twins by providing users with seamless, automated, and transparent access to datasets, this system would be a first step towards this goal by allowing them to interact with the data efficiently and intuitively.

Beyond system-level considerations, an important direction for future work is to further optimise the algorithm in order to improve performance and scalability to very large datasets. This includes parallelisation, by running point-in-polygon tests at the quadtree leaf nodes concurrently for example. Recent work has also shown that quadtree construction and traversal can be parallelised [10], and exploring how such methods could be integrated into our algorithm may result in significant performance improvements. In addition, GPU-optimised quadtree traversal [7] represents another promising research avenue. Besides parallelisation, further optimisations could involve exposing the quadtree compression level as a user-controlled parameter and studying how optimal compression varies with grid types and resolutions. Finally, while the algorithm was designed to operate on fully unstructured grids, many real-world datasets exhibit at least partial structure. Future work could therefore investigate how this structure, such as that present in Lambert or other projected grids, can be exploited to further optimise performance by using, for example, more specialised quadtree variants or alternative spatial data structures in our algorithm.

Acknowledgments

The authors would like to thank Professor Stefan Wesner for helpful discussions about this work.

The work presented in this paper has been produced in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] Kunio Aizawa and Shojiro Tanaka. 2008. A constant-time algorithm for finding neighbors in quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 7 (2008), 1178–1183.

[2] Marius Almanstötter, Tobias Melson, Hans-Thomas Janka, and Ewald Müller. 2018. Parallelized solution method of the three-dimensional gravitational potential on the Yin–Yang grid. *The Astrophysical Journal* 863, 2 (2018), 142.

[3] Cédric Bastoul and Paul Feautrier. 2003. Improving data locality by chunking. In *International Conference on Compiler Construction*. Springer, 320–334.

[4] Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 9 (1975), 509–517.

[5] Stefan Berchtold, Daniel A Keim, and Hans-Peter Kriegel. 1996. The X-tree: An index structure for high-dimensional data. (1996).

[6] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. 2022. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv preprint arXiv:2211.02556* (2022).

[7] Wade Brainerd, Theresa Foley, Manuel Kraemer, Henry Moreton, and Matthias Nießner. 2016. Efficient GPU rendering of subdivision surfaces using adaptive quadtrees. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

[8] Ramon De Elia, René Laprise, and Bertrand Denis. 2002. Forecasting skill limits of nested, limited-area models: A perfect-model approach. *Monthly Weather Review* 130, 8 (2002), 2006–2023.

[9] Charles H Deetz. 1918. The Lambert conformal conic projection. *US Department of Commerce, US Coast and Geodetic Survey, Special Publication* 47 (1918), 1–61.

[10] Landon Dyken, Will Usher, Steve Petruzza, Stavros Sintos, and Sidharth Kumar. 2025. Accelerating Web-Based Graph Drawing with Bottom-Up GPU Quadtree Construction. In *2025 IEEE 18th Pacific Visualization Conference (PacificVis)*. IEEE, 24–29.

[11] ECMWF. 2024. GribJump Software. https://github.com/ecmwf/gribjump Accessed on 2025-11-08.

[12] Ahmed Eldawy, Louai Alarabi, and Mohamed F Mokbel. 2015. Spatial partitioning techniques in SpatialHadoop. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1602–1605.

[13] European Centre for Medium-Range Weather Forecasts (ECMWF). 2025. Time–averaged geographical mean for 2 m temperature (4DVAR). ECMWF Charts / OBSTAT catalogue. https://charts.ecmwf.int/catalogue/packages/obstat/ Dataset accessed: https://charts.ecmwf.int/catalogue/packages/obstat/ (last accessed 2025-11-08).

[14] Thomas Geenen, Nils Wedi, Sebastian Milinski, Ioan Hadade, Balthasar Reuter, Simon Smart, James Hawkes, Emma Kuwertz, Tiago Quintino, Emanuele Danovaro, et al. 2024. Digital twins, the journey of an operational weather prediction system into the heart of Destination Earth. *Procedia Computer Science* 240 (2024), 99–109.

[15] Krzysztof M Gorski, Eric Hivon, Anthony J Banday, Benjamin D Wandelt, Frode K Hansen, Mstvos Reinecke, and Matthia Bartelmann. 2005. HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal* 622, 2 (2005), 759.

[16] Erik W Grafarend. 1995. The optimal universal transverse Mercator projection. *Manuscripta geodaetica* 20, 6 (1995), 421–468.

[17] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. 47–57.

[18] Eric Haines. 1994. Point in Polygon Strategies. *Graphics Gems* 4, 1 (1994), 24–46.

[19] Jianqiang Hao, Jianzhi Sun, Yi Chen, Qiang Cai, and Li Tan. 2018. Optimal reliable point-in-polygon test and differential coding boolean operations on polygons. *Symmetry* 10, 10 (2018), 477.

[20] Paul S Heckbert and Pat Hanrahan. 1984. Beam tracing polygonal objects. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 119–127.

[21] Kelsey Jordahl, Joris Van Den Bossche, Martin Fleischmann, James McBride, Jacob Wasserman, Matt Richards, Adrian Garcia Badaracco, Alan D. Snow, Jeffrey Gerard, Jeff Tratner, Matthew Perry, Brendan Ward, Carson Farmer, Geir Arne Hjelle, Micah Cochran, Mike Taves, Sean Gillies, Giacomo Caria, Lucas Culbertson, Matt Bartos, Nick Eubank, Ray Bell, Sangarshanan, John Flavin, Sergio Rey, Maxalbert, Aleksey Bilogur, Christopher Ren, Dani Arribas-Bel, and Daniel Mesejo-León. 2022. *geopandas/geopandas: v0.12.1.* doi:10.5281/zenodo.7262879

[22] Akira Kageyama and Tetsuya Sato. 2004. "Yin-Yang grid": An overset grid in spherical geometry. *Geochemistry, Geophysics, Geosystems* 5, 9 (2004).

[23] Daniel Klocke, Claudia Frauen, Jan Frederik Engels, Dmitry Alexeev, René Redler, Reiner Schnur, Helmuth Haak, Luis Kornblueh, Nils Brüggemann, Fatemeh Chegini, et al. 2025. Computing the Full Earth System at 1km Resolution. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 125–136.

[24] Moritz Laass. 2021. Point in Polygon Tests Using Hardware Accelerated Ray Tracing. In *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 666–667.

[25] Simon Lang, Mihai Alexe, Matthew Chantry, Jesper Dramsch, Florian Pinault, Baudouin Raoult, Mariana CA Clare, Christian Lessig, Michael Maier-Gerber, Linus Magnusson, et al. 2024. AIFS–ECMWF's data-driven forecasting system. *arXiv preprint arXiv:2406.01465* (2024).

[26] Mathilde Leuridan, Christopher Bradley, James Hawkes, Tiago Quintino, and Martin Schultz. 2025. Performance Analysis of an Efficient Algorithm for Feature Extraction from Large Scale Meteorological Data Stores. In *Proceedings of the Platform for Advanced Scientific Computing Conference*. 1–9.

[27] Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, Martin Schultz, and Tiago Quintino. 2025. Polytope: an algorithm for efficient feature extraction on hypercubes. *Journal of Big Data* 12, 1 (2025), 1–25.

[28] Mathilde Leuridan, Adam Warde, Oisin-M, James Hawkes, James Varndell, Peter Tsrunchev, Chris Bradley, Dušan Figala, Iain Russell, Kai Kratz, Simon Smart, and Yordan Radev. 2025. *ecmwf/polytope: 2.1.1.* doi:10.5281/zenodo.17483028

[29] Lingcheng Li, Gautam Bisht, Dalei Hao, and L Ruby Leung. 2024. Global 1 km land surface parameters for kilometer-scale Earth system modeling. *Earth System Science Data* 16, 4 (2024), 2007–2032.

[30] Xiaolan Li, Yongqiang Yu, Hailong Liu, and Pengfei Lin. 2017. Sensitivity of Atlantic meridional overturning circulation to the dynamical framework in an ocean general circulation model. *Journal of Meteorological Research* 31, 3 (2017), 490–501.

[31] King-Ip Lin, Hosagrahar V Jagadish, and Christos Faloutsos. 1994. The TV-tree: An index structure for high-dimensional data. *The VLDB Journal* 3, 4 (1994), 517–542.

[32] Leonidas Linardakis, Daniel Reinert, and Almut Gassmann. 2011. *Icon grid documentation.* Technical Report. Tech. rep., DKRZ, Hamburg, available at: https://www. dkrz. de/SciVis/hd-cp ….

[33] Richard J Lisle and Peter R Leyshon. 2004. *Stereographic projection techniques for geologists and civil engineers.* Cambridge University Press.

[34] Gurvan Madec et al. 2015. NEMO ocean engine. (2015).

[35] Gurvan Madec and Maurice Imbard. 1996. A global ocean mesh to overcome the North Pole singularity. *Climate Dynamics* 12, 6 (1996), 381–388.

[36] Sylvie Malardel, Nils Wedi, Willem Deconinck, Michail Diamantakis, Christian Kühnlein, George Mozdzynski, Mats Hamrud, and Piotr Smolarkiewicz. 2016. A new grid for the IFS. *ECMWF newsletter* 146, 23-28 (2016), 321.

[37] Donald Meagher. 1982. Geometric modeling using octree encoding. *Computer graphics and image processing* 19, 2 (1982), 129–147.

[38] Alistair Miles, jakirkham, Joe Hamman, Dimitri Papadopoulos Orfanos, David Stansby, M Bussonnier, Josh Moore, Davis Bennett, Tom Augspurger, Norman Rzepka, Deepak Cherian, Sanket Verma, James Bourbeau, Andrew Fulton, Ryan Abernathey, Gregory Lee, Hannes Spitz, Mads R. B. Kristensen, Max Jones, Zain Patel, Saransh Chopra, Matthew Rocklin, AWA BRANDON AWA, Nathan Zimmerman, Martin Durant, Juan Nunez-Iglesias, Elliott Sales de Andrade, and Vincent Schut. 2025. *zarr-developers/zarr-python: v2.18.5.* doi:10.5281/zenodo.15101732

[39] Ken Museth, Jeff Lait, John Johanson, Jeff Budsberg, Ron Henderson, Mihai Alden, Peter Cucka, David Hill, and Andrew Pearce. 2013. OpenVDB: an open-source data structure and toolkit for high-resolution volumes. In *Acm siggraph 2013 courses*. 1–1.

[40] Stig Nordbeck and Bengt Rystedt. 1967. Computer cartography point-in-polygon programs. *BIT Numerical Mathematics* 7, 1 (1967), 39–64.

[41] Regina Obe and Leo Hsu. 2021. *PostGIS in action.* Simon and Schuster.

[42] P Oddo and Nadia Pinardi. 2008. Lateral open boundary conditions for nested limited area models: A scale selective approach. *Ocean modelling* 20, 2 (2008), 134–156.

[43] Cédric Penard, Flavien Gouillon, Xavier Delaunay, and Sylvain Herlédan. 2025. A new sub-chunking strategy for fast netCDF-4 access in local, remote and cloud infrastructures, chunkindex V1. 1.0. *EGUsphere* 2025 (2025), 1–24.

[44] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. 2023. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796* (2023).

[45] The GeoRust Project. 2025. *geo: geospatial primitives and algorithms.* GeoRust. https://crates.io/crates/geo Rust crate providing geospatial primitives and algorithms.

[46] Burkhard Rockel, Andreas Will, and Andreas Hense. 2008. The regional climate model COSMO-CLM (CCLM). *Meteorologische zeitschrift* 17, 4 (2008), 347.

[47] Radu Bogdan Rusu and Steve Cousins. 2011. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*. IEEE, 1–4.

[48] Hanan Samet. 1982. Neighbor finding techniques for images represented by quadtrees. *Computer graphics and image processing* 18, 1 (1982), 37–57.

[49] Hanan Samet. 1984. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)* 16, 2 (1984), 187–260.

[50] Hans Segura, Xabier Pedruzo-Bagazgoitia, Philipp Weiss, Sebastian K Müller, Thomas Rackow, Junhong Lee, Edgar Dolores-Tesillos, Imme Benedict, Matthias Aengenheyster, Razvan Aguridan, et al. 2025. nextGEMS: entering the era of kilometer-scale Earth system modeling. *Geoscientific Model Development* 18, 20 (2025), 7735–7761.

[51] Moshe Shimrat. 1962. Algorithm 112: position of point relative to polygon. *Commun. ACM* 5, 8 (1962), 434.

[52] Hoang Vo, Ablimit Aji, and Fusheng Wang. 2014. SATO: a spatial data partitioning framework for scalable query processing. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*. 545–548.

[53] Tin Vu and Ahmed Eldawy. 2020. R*-grove: Balanced spatial partitioning for large-scale datasets. *Frontiers in big data* 3 (2020), 28.

[54] Ning Wang and Jin-Luen Lee. 2011. Geometric properties of the icosahedral-hexagonal grid on the two-sphere. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2536–2559.

[55] Yong Wang, Zhenling Liu, Hongyan Liao, and Chengjun Li. 2015. Improving the performance of GIS polygon overlay computation with MapReduce for spatial big data processing. *Cluster Computing* 18, 2 (2015), 507–516.
[56] Nigel P Weatherill. 1999. Unstructured Grids.
[57] RT Williams. 1995. Lambert and Mercator map projections in geology and geophysics. *Computers & Geosciences* 21, 3 (1995), 353–364.
[58] Sheng Yang, Jun-Hai Yong, Jiaguang Sun, Hejin Gu, and Jean-Claude Paul. 2010. A point-in-polygon method based on a quasi-closest point. *Computers & Geosciences* 36, 2 (2010), 205–213.
[59] Borut Žalik and Ivana Kolingerova. 2001. A cell-based point-in-polygon algorithm suitable for large sets of points. *Computers & Geosciences* 27, 10 (2001), 1135–1145.
[60] Günther Zängl, Daniel Reinert, Pilar Rípodas, and Michael Baldauf. 2015. The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. *Quarterly Journal of the Royal Meteorological Society* 141, 687 (2015), 563–579.
[61] Jianting Zhang and Simin You. 2012. Speeding up large-scale point-in-polygon test based spatial join on GPUs. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*. 23–32.

7 Beyond Standard Datacubes

We have now established a robust framework for feature extraction on top of standard, well-defined datacubes. This framework supports a wide range of use cases and provides significant improvements over traditional data services, which typically require reading and transferring large volumes of data to satisfy user requests. In contrast, the feature extraction approach developed here minimises unnecessary data access and transfer.
This framework performs well for standard datacubes in which datasets are relatively homogeneous. In practice, however, scientific datasets often differ from one another in non-trivial ways. As we move through scientific data spaces, datasets may exhibit specific relational constraints that influence how they are organised and accessed. In particular, a data space may display branching behaviour or impose different constraints depending on the chosen perspective. These characteristics highlight the limitations of the traditional datacube abstraction.
In the following paper [73], we address these limitations by introducing a more general data representation, which we refer to as data hypercubes. This abstraction models data spaces as maximally compressed tree-based structures with dense datacubes at the leaves, enabling efficient representation of complex data without resorting to an overly broad superset of the available data. We motivate the abstraction, define its core concepts, express standard datacube operations within this framework and evaluate one of its concrete implementations through the Qubed software. Finally, we integrate the Qubed data hypercube model into the Polytope data extraction service to enable feature extraction over arbitrary complex NWP data spaces and investigate the resulting unified extraction system.

This work is currently in preparation for submission to a peer-reviewed journal and the open-source implementation of the work described here is available on GitHub as part of the Polytope repository [75]. I am the primary author of this publication. The specific contributions of myself and my co-authors are outlined below according to the Contributor Roles Taxonomy (CRediT):

M Leuridan: Conceptualisation, Software, Formal analysis, Visualisation, Writing-original draft; *J. Hawkes*: Conceptualisation, Writing-review & editing; *M. Schultz*: Writing-review & editing; *T. Quintino*: Conceptualisation, Writing-review & editing.

Beyond Standard Datacubes: Extracting Features from Irregular and Branching Earth System Data

Mathilde Leuridan James Hawkes Tiago Quintino Martin Schultz

February 20, 2026

Abstract

Earth science datasets are growing rapidly in both volume and structural complexity. They increasingly contain richly labelled data with heterogeneous metadata and complex internal constraints that impose dependencies between variables and dimensions. Datacubes have become a common abstraction for organising such datasets, but traditional dense and orthogonal datacube models struggle to represent irregular, sparse or branching data spaces efficiently. In particular, gaps in coverage, conditional dimensions and heterogeneous variable definitions are poorly captured by existing approaches. In this paper, we introduce a generalised data hypercube representation based on compressed tree structures, which enables an accurate and compact description of complex data spaces. We describe the design of this representation and analyse its ability to capture sparsity and conditional relationships while remaining efficient to traverse. Using a concrete implementation, we study the performance characteristics of compressed tree data hypercubes and demonstrate their effectiveness as fast, cache-like indices over large backend data stores. Building on this representation, we present an integrated feature extraction system that operates directly on tree-based data hypercubes within the Polytope framework. By embedding data access strategies into the data hypercube abstraction itself, the system enables precise, sub-field data extraction and supports flexible, user-driven access patterns. We evaluate the performance of the integrated system and show how it enables new ways of interacting with complex datasets that are difficult to support using traditional access models. This work bridges the gap between expressive data hypercube models and efficient data access methods. In particular, it provides a unified framework that combines tree-based data representations with feature extraction capabilities. The proposed approach therefore offers a foundation for scalable and user-centric access to large heterogeneous Earth science datasets and highlights directions for future development and deployment.

1 Introduction

Advances in the fields of earth observation, climate modelling and numerical weather prediction have led to unprecedented volumes of geospatial and meteorological data [12, 39, 18]. Modern satellites generate a continuous stream of observations across hundreds of spectral channels and instruments, while weather and climate models produce increasingly detailed forecasts with multiple ensemble members, parameterisations and output frequencies. Foundational standardisation efforts, such as the netCDF data model [41] that is implemented with the Climate and Forecast (CF) [8] and Climate Model Output Rewriter (CMOR) [35] metadata conventions, have enabled efficient representation and exchange of datasets structured primarily along the familiar latitude, longitude, vertical level and time dimensions. However, modern datasets increasingly

extend beyond this standard model, introducing additional axes such as height levels, instrument modes or forecast lead times, and thereby creating richly structured yet highly complex data spaces.

Whilst this abundance of data offers scientific opportunities, it also creates significant practical challenges [47, 19]. Indeed, high-dimensional datasets can be difficult to visualise, reason about and manipulate. Analytical operations that are trivial in low-dimensional settings, such as slicing out subsets, combining variables or mapping between coordinate systems, quickly become cumbersome when the data spans irregular grids, contains conditional dependencies or has gaps arising from instrument geometry or quality control. Beyond representation alone, these difficulties directly impact the ability to filter, query and extract meaningful subsets or features from the data, which are essential steps in most scientific workflows. As a consequence, there is a growing need for computational frameworks that can represent such datasets uniformly, while simultaneously enabling efficient and intuitive data access.

One key response to this need has been the development of the datacube abstraction. First originating from the business analytics community, the datacube concept provided a way to model multidimensional tabular data and to support operations such as slicing and pivoting [17, 20, 36]. Over the last decade, this idea has been reinterpreted and expanded within the earth sciences community to accommodate raster and observational datasets [25, 31, 16, 15]. Efforts such as RasDaMan [4, 3] and ESA's Deep Earth System Data Laboratory (DeepESDL) [2, 6] have in particular demonstrated how datacubes can unify heterogeneous datasets and simplify access to complex archives. However, most implementations primarily focus on organising and exposing the structure and metadata of homogeneous multidimensional data, while higher-level data filtering and feature extraction are often treated as downstream, ad-hoc processes.

In the open-source scientific ecosystem, the xarray library [22] has further popularised the datacube concept by providing a labeled tensor representation for multidimensional arrays. In meteorology, climate research and remote sensing, xarray has become central to data analysis workflows because it facilitates essential operations on complex datasets. The tensor model underpinning xarray carries a few implicit assumptions however. Amongst them, it assumes that data lies on orthogonal, regularly spaced coordinate axes and forms dense arrays without structural gaps. These assumptions are increasingly strained by modern datasets, which may be sparse, irregular or defined only for particular combinations of dimensions, such as when variables depend on specific instruments or ensemble configurations. In such cases, even standard filtering or extraction tasks may require substantial preprocessing or specific logic outside of the datacube abstraction itself.

A potential solution to this is to split data into multiple lower-dimensional datacubes. DeepESDL, for example, stores each geophysical parameter as its own 3- or 4-dimensional cube. However, whilst this is a practical representation, this fragmentation hides relationships between parameters and prevents a unified view of the dataset as a whole. More importantly, it complicates the implementation of generic data extraction workflows that must operate consistently across parameters, dimensions and conditional dependencies. Recognising these limitations, xarray recently introduced DataTrees [49, 37], a hierarchical extension designed to express related datacubes in a tree-like structure. DataTrees offer greater flexibility by allowing branching and grouping, but they still impose structural uniformity. Indeed, branches must share similar dimensional assumptions, coordinate ordering and overall data hierarchy, which limits their ability to support generic filtering and feature extraction across highly irregular data spaces.

In this paper, we consider a more general representation of high-dimensional data spaces based on a flexible hierarchical tree structure, designed to capture the coordinate structure of complex, heterogeneous and irregular datasets beyond the constraints of existing datacube frameworks. By encoding dimensional dependencies and structural rules directly in the tree, this approach

relaxes assumptions of regularity, completeness and uniformity, while supporting efficient operations such as querying, merging, filtering and traversal even for sparse or non-orthogonal data spaces. Crucially, this representation is intended not only to describe the data, but also to serve as a foundation for end-to-end data extraction workflows.
We then briefly describe the Qube [10] as a concrete realisation of this compressed tree-like datacube representation, and investigate its performance and potential use in practical applications. In particular, we present a prototype integration of the Qube within the Polytope feature extraction framework [29, 28], illustrating how feature extraction can be implemented directly on top of generic datacube representations. This enables a complete workflow in which data representation, filtering and feature extraction are tightly integrated, without imposing restrictive assumptions on the structure of the underlying data. Finally, we discuss this integrated framework, highlighting the benefits of a more general approach to datacubes and its potential applicability across a wide range of data extraction and analysis systems.

2 Motivation and Background

In recent years, considerable work has focused on formalising the concept of datacubes and multidimensional data spaces to better accommodate the growing complexity of geospatial datasets. In this section, we motivate the need for such abstractions and review current implementations, considering both how multidimensional data are represented and how they are accessed and used in practice. We discuss why high-dimensional scientific datasets require more systematic representations and examine prior frameworks that have shaped current approaches, drawing throughout on examples from meteorological and climate data. In particular, we highlight how characteristics such as multi-instrument observations, ensemble forecasts, heterogeneous grids and conditional data availability stress traditional datacube assumptions. We also review the data access strategies used in existing frameworks, including querying, subsetting and filtering mechanisms, and examine how these choices affect usability and performance for weather and climate applications. Together, these considerations motivate the need for more flexible and unified datacube structures that support both expressive representations, as well as efficient data access.

2.1 Datacube Abstraction and Data Representations

The datacube concept was first introduced in the field of business analytics as a way to organise and efficiently query multidimensional tabular databases [20]. Earlier datacube models supported operations such as aggregation, slicing and pivoting across different axes like product, region and time, providing a unified abstraction for structured multidimensional data [17]. As data-intensive scientific disciplines expanded, this abstraction proved broadly applicable beyond its original domain. Many fields, including biomedical imaging, genomics, environmental monitoring, social sciences and large-scale sensor networks, routinely generate datasets that can be described along multiple coordinate dimensions [26, 24, 51]. Over time, the datacube has therefore evolved into a widely adopted conceptual model for managing complex, high-dimensional datasets in a structured and interpretable form.
This evolution is reflected in scientific software frameworks such as xarray [22] and storage formats such as Zarr [50], which represent datasets as labeled tensors or multidimensional arrays with named coordinates. These tensor-based models have become foundational in many data-driven workflows because they support intuitive indexing, efficient numerical computation and integration with high-performance computing and machine learning ecosystems [11, 45, 27]. For datasets that are dense, uniform and defined on orthogonal coordinate axes, such as regularly

gridded simulations, standardised imaging data or homogeneous observational records, these representations provide both conceptual clarity and practical efficiency.
However, many real-world datasets deviate substantially from these assumptions. Across domains, data may be sparse, heterogeneous or defined only for particular combinations of variables. In medicine, patient records often contain missing values, condition-specific measurements or varying record lengths. Genomic datasets exhibit irregular coverage and alignment. Sensor network data may be intermittent or constrained by device-specific configurations. Meteorological and Earth system datasets, in particular, exhibit many of these challenges simultaneously. Climate simulations typically produce a single long time series, while weather forecasts introduce both an initialisation time and a forecast lead time, resulting in multiple temporal axes. Earth observation products add further dimensions related to instrument mode, orbit geometry or viewing configuration. As a result, no single dense, orthogonal tensor can faithfully represent the combined data space.
Within meteorological and climate datasets, further structural variability arises. Different variables may be available at different forecast lead times, vertical coordinates may be defined on model levels, pressure levels or only at the surface, and observational data may exist only under specific quality constraints or sensor configurations. This leads to branching data spaces in which dimensionality, coordinate definitions and data availability vary across subsets of the dataset. Although many subspaces share common structural patterns, the overall data space becomes irregular and sparse, with conditional dependencies that are difficult to express in traditional tensor models.
To address some of these limitations, xarray introduced DataTrees [49, 37], a hierarchical abstraction in which related datacubes are organised into a tree structure. This allows groups of datasets to share common dimensions while permitting branching to represent subsets with different structures. Tree-based datacube models represent an important step toward capturing relational and conditionally defined data spaces, and they are particularly relevant for organising complex Earth system datasets composed of multiple related products.
Nevertheless, existing hierarchical datacube models still impose significant structural constraints. They typically assume fixed axis orderings, homogeneous dimensionality across sibling branches and a rigid notion of what constitutes a valid subspace. They struggle to represent datasets in which dimensionality itself varies across the domain, where different branches require different coordinate sets, or where successive constraints dynamically change the relevant dimensions. In such cases, the structure of the data cannot be expressed naturally without fragmentation or duplication. These limitations motivate the need for more general and flexible representations that can encode heterogeneous, sparse and structurally variable data spaces while preserving shared structure and enabling efficient navigation.

2.2 Datacube Access and Data Extraction

Beyond pure representation, an equally critical aspect of working with multidimensional datasets is how data is accessed and extracted from the underlying data stores. In practice, users rarely require entire datacubes. Instead, they seek specific subsets corresponding to regions of interest, time windows, vertical levels, ensemble members or combinations of conditional constraints. Effective datacube frameworks must therefore support data access strategies that allow users to retrieve only the information relevant to their scientific question, both efficiently and intuitively.
Traditional datacube frameworks support access through operations such as slicing, indexing and aggregation along coordinate axes [17, 46]. In tensor-based models, these operations are often tightly coupled to array indexing and chunking strategies [38, 32], which work well for regular, dense datasets. However, as data spaces become more irregular and conditionally structured, these access patterns become increasingly difficult to generalise. Querying sparse or branching

data often requires custom logic, preprocessing or manual traversal of multiple datacubes, reducing usability and increasing the computational cost of data access.
In response to these challenges, the concept of feature extraction has gained increasing prominence in meteorology and climate science as a higher-level data access strategy. Rather than exposing raw multidimensional arrays, feature extraction frameworks allow users to specify the subsets of data they are interested in, such as trajectories, cross-sections, vertical profiles or spatial regions, directly in terms of scientific features. This shift is reflected in the development of interfaces such as the OGC Environmental Data Retrieval (EDR) standard [5], which enables users to request specific features from complex geospatial datasets through a unified API.
At the institutional level, similar ideas have emerged in systems such as the Polytope feature extraction framework at ECMWF [29, 28], which allows users to describe complex extraction requests over high-dimensional weather and climate datasets. Related approaches are also present in data services built on platforms such as RasDaMan [4, 3] or the Sentinel Hub [34], where users can request spatial or temporal subsets through higher-level clipping and subsetting operations. These developments reflect the shift towards feature extraction as an essential way of interacting with modern datacubes and data systems.
However, many existing datacube implementations treat feature extraction as a post-processing step applied after data have already been retrieved from backend storage. In such systems, large volumes of data are first accessed and transferred before being clipped or post-processed to produce the final user output. Whilst this approach is often sufficient for smaller datasets, it becomes increasingly inefficient for weather and climate data, where backend archives can span petabytes and user requests typically target only a few megabytes of information. In such applications, the cost of accessing unnecessary data dominates the workflow, limiting scalability and performance.
This gap between datacube abstractions and efficient data access strategies points to the need for new datacube frameworks. In such frameworks, feature extraction should be treated as a key datacube operation and tightly integrated with the underlying data model. Access strategies should be able to exploit knowledge of the data structure itself, allowing selective traversal and retrieval of only the relevant subspaces. Bridging representation and feature-based data access in this way is essential for enabling scalable and user-friendly workflows in meteorology, climate science and other data-intensive disciplines.

3 A Datacube Generalisation

Traditional datacube models assume that data occupy a dense, orthogonal and fully defined multidimensional space. As shown in the previous sections, these assumptions increasingly break down for modern scientific datasets, particularly in meteorology and climate science, where data availability depends on conditional constraints, dimensionality varies across subsets and large portions of the space may be sparse or undefined. Representing such datasets as complete datacubes either leads to fragmentation into multiple disconnected cubes or to inefficient padding with missing values.
To address these limitations, we introduce a generalisation of the datacube concept that relaxes global assumptions of completeness and orthogonality. Instead of viewing a datacube as a single multidimensional array, we represent the data space as a compressed hierarchical structure that explicitly encodes branching and conditional relationships. This representation, which we refer to as a data hypercube, preserves shared structure where possible, while allowing divergence where required. It therefore provides a natural foundation for efficient querying and feature-oriented data access. In the remainder of this section, we describe this generalised datacube representation, formalise its structure and discuss the advantages it offers in terms of

expressiveness, performance and reuse across meteorological data systems.

3.1 Data Hypercubes

The generalised data hypercube represents multidimensional data spaces as compressed tree structures instead of dense datacubes. Each level of the tree corresponds to a dimension and branching represents conditional structure due to data constraints. Paths from the root to a leaf define valid combinations of coordinates for which data exists.

In practice, many geophysical datasets are characterised by constraints between dimensions, meaning that not all combinations of coordinates are meaningful or populated. For example, consider a meteorological dataset containing a set of atmospheric variables. Some variables, such as 2-metre temperature or surface pressure, are defined only at the surface, while others, such as temperature, wind or humidity, are defined on multiple pressure levels. The vertical dimension is therefore conditional on the variable, and treating all variables as sharing the same set of vertical coordinates would either require introducing artificial levels for surface-only variables or separating the dataset into multiple independent datacubes, thereby obscuring relationships between variables. Instead, the generalised data hypercube tree formulation allows dense, orthogonal sub-datacubes to be represented efficiently, while also accommodating sparse, heterogeneous and irregular data spaces.

Formally, a data hypercube can be defined as a rooted, directed tree

$$\mathcal{T} = (V, E),$$

where each node $v \in V$ is associated with a dimension $d(v)$ and a subset of admissible coordinate values $C_v \subseteq \mathcal{C}_{d(v)}$. An edge $(v_i, v_j) \in E$ represents refinement by an additional constraint on a subsequent dimension. Each root-to-leaf path therefore defines a valid subspace of the overall domain, and leaf nodes can then be associated with payloads such as data arrays, storage references or various metadata. In this context, fully dense datacubes correspond to flat trees where each dimension level exhaustively enumerates its coordinate domain.

In the generalised representation, the structural variability in the example above is expressed through branching in the hypercube tree. Higher levels of the tree may encode shared dimensions such as forecast lead time, while a subsequent branching distinguishes between surface variables and pressure-level variables. One branch terminates at the surface, while the other introduces an additional vertical dimension corresponding to pressure levels. Each leaf node therefore represents an internally consistent subspace with a well-defined dimensional structure. Shared dimensions are represented once, and divergence is introduced only where the data structure genuinely differs.

Unlike classical datacubes, this representation is not invariant under permutation of dimensions. The order in which dimensions appear along the tree defines both the logical structure of the data space, as well as the optimal traversal order used for querying. Dimensions higher up in the tree act as early filters, which allows for pruning of entire subspaces when they are irrelevant to a given request. Selecting an appropriate dimension ordering therefore becomes an important design choice that can optimally be aligned with common access patterns. In meteorological and climate workflows, this may be implemented by placing higher-level distinctions, such as different satellite instruments or meteorological model types, towards the top of the tree, so that branches corresponding to incompatible acquisition systems or modelling frameworks can be excluded early in the traversal. Once the data space has been partitioned along these distinctions, the remaining dimensions within each resulting sub-hypercube are typically more homogeneous, allowing for more efficient querying and internal organisation within each branch.

This data hypercube representation provides a compact and expressive way to model complex

meteorological data spaces. By explicitly encoding structural dependencies, it enables both efficient traversal as well as robust filtering, forming the basis for more advanced data extraction and analysis pipelines.
Tree-based datacube representations of this form have already been adopted in different operational contexts at ECMWF. They underpin the internal representation of data availability in the MARS archive [40] and are also used as the backend of the Destination Earth STAC catalogue [1] through the Qubed software [10]. In these systems, data hypercubes have demonstrated their ability to efficiently encode large, heterogeneous meteorological data stores while supporting scalable data discovery and access.

3.2 Data Hypercube Operations

The data hypercube representations introduced previously are built on compressed tree structures that support a well-defined set of operations. These operations are essential both for constructing data hypercubes efficiently and for maintaining their optimality as the underlying data evolve over time. In practice, data hypercube trees must be built incrementally, recomputed as new data become available, and traversed efficiently in response to diverse user queries. A key part of this are the operations on the tree structure itself. Basic operations include traversal, filtering and slicing along dimensions, which enable selective exploration of the data space.
From a construction perspective, consider a data hypercube defined over a set of dimensions $D = \{d_1, \ldots, d_k\}$, where each dimension d_i has cardinality n_i. In practice, the resulting tree representation is not necessarily uniform. Indeed, due to sparsity or missing coordinate combinations, different branches of the tree may terminate at different tree depths. Let $k_{\max}$ denote the maximum depth of any root-to-leaf path in the tree. A naïve construction procedure inserts each available data tuple independently by creating a path through the tree corresponding to its coordinates. If the underlying dataset contains N such tuples, which implies that N is the number of leaf nodes, then each insertion requires at most traversing or creating a path of length $k_{\max}$. The total construction cost is therefore bounded by

$$T^{\text{naive}}_{\text{construct}} = \mathcal{O}(k_{\max} N).$$

Similarly, extraction or traversal operations in this representation may require visiting a large fraction of the N leaf nodes, and set operations such as unions or intersections between two data hypercubes with N_1 and N_2 leaves require

$$T^{\text{naive}}_{\cup}, T^{\text{naive}}_{\cap} = \mathcal{O}(N_1 + N_2).$$

More advanced set operations, such as unions and intersections of trees, are required to combine multiple datasets, reconcile overlapping data spaces or restrict a representation to shared subsets of dimensions and coordinates. These operations make it possible to construct complex data spaces successively from simpler ones. Union operations play a central role when integrating heterogeneous datasets or aggregating data over time, for example when successive forecast cycles or observational products are added to an existing data store. In such cases, multiple data hypercube trees must be merged into a single representation that preserves shared structure while introducing branching only where the data differs. In contrast, intersection operations serve the purpose of restricting a data hypercube to a common subspace, such as when identifying overlapping availability between datasets or applying constraints derived from user queries.
To ensure efficiency, these set operations must be accompanied by compression and re-optimisation of the tree structure. After unions or intersections, the resulting tree might contain redundant branches or uncompressed duplicated subtrees that degrade traversal performance. Let compression collapse structurally identical subtrees so that a tree with N nodes is reduced to a

representation with only

$$M \ll N$$

distinct structural nodes.
The compression operation itself consists of identifying structurally identical subtrees and replacing them with a shared representation. This requires comparing branches of the tree that span the same coordinate subspaces in order to detect similarities between sub-hypercubes. In the worst case, this comparison must be performed across all nodes of the uncompressed representation, so that the compression step is linear in the number of nodes,

$$T_{\text{compress}} = \mathcal{O}(N).$$

In practice, however, compression is normally performed bottom-up, beginning at the leaves of the tree and propagating upwards. Since identical subtrees are more likely to occur towards the leaves of the tree, performing compression in this way reduces the number of distinct intermediate nodes that need to be compared higher up in the tree. As a result, tree hierarchies that maximise redundancy in deeper layers not only decrease the final number of structural nodes M, but also reduce the effective cost of the compression procedure itself.
After compression, subsequent construction and transformation operations can then be performed in terms of these unique sub-hypercubes rather than the full Cartesian space, yielding

$$T_{\text{construct}}^{\text{compressed}} = \mathcal{O}(k_{\max} M), \qquad T_{\cup}^{\text{compressed}}, T_{\cap}^{\text{compressed}} = \mathcal{O}(M_1 + M_2).$$

Hence unions or intersections between compressed data hypercubes no longer require traversing all N nodes, but only the M distinct structural nodes, resulting in a reduction in computational cost by a factor proportional to $(N_1 + N_2)/(M_1 + M_2) \gg 1$.
Compression is therefore an essential operation as it collapses identical subtrees and factors out shared tree structure, restoring an efficient data hypercube compressed tree object after union, intersection or any other transformation. As a result, compression is an integral part of the data hypercube construction process. It ensures that the resulting representation remains compact and well aligned with typical access patterns. It also highlights that the achievable efficiency depends directly on how the dimensions are arranged in the tree hierarchy as dimension orderings that maximise shared sub-hypercubes minimise M, whereas early branching along highly variable dimensions increases M and degrades the performance of construction, extraction and set operations.

3.3 Data Hypercube Performance

The performance of data hypercube representations is a central concern in any practical data workflow. Operations such as construction, compression, union and intersection are fundamental steps in building, updating and maintaining coherent representations of complex data spaces. As these operations are repeatedly applied throughout the lifetime of a data hypercube, their computational characteristics directly influence the scalability and performance of downstream applications and data access. Understanding how efficiently these operations can be performed is therefore essential for assessing the practical use of such generalised data hypercube frameworks. Performance considerations are especially important in operational and time-critical settings, where data spaces evolve continuously and up-to-date representations must be generated and traversed with minimal latency. In such contexts, inefficiencies in hypercube construction or traversal can quickly start to dominate the overall system cost, limiting the ability to serve timely and consistent data products.

3.3.1 Data Workflows

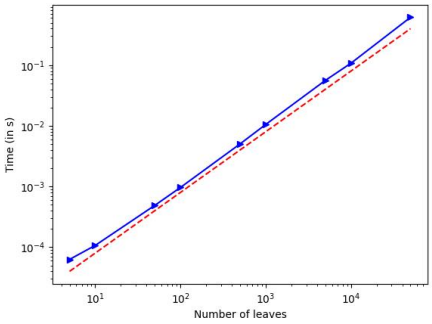
In the following, we study performance using Qubes [10] as a concrete implementation of compressed tree-based data hypercube representations. While the analysis focuses on Qubes, the results are representative of a broader class of generalised data hypercube frameworks that encode structure, sparsity and conditional relationships using hierarchical, compressed trees.
Within the workflows considered here, Qubes act primarily as fast indices and caches over large underlying data stores. Their effectiveness therefore depends not only on their expressive power, but also on how efficiently they can be constructed, maintained and reused as the underlying data change. Understanding the performance behaviour of core Qube operations is thus central to evaluating their suitability for operational use.
We analyse the performance of Qubes by considering their lifecycle within a typical data workflow. We first examine the cost of Qube construction, which represents the entry point to the data structure. We then study compression, focusing on its computational characteristics and its role in maintaining efficiency at scale. Compression introduces an inherent trade-off as, although it increases the cost of construction, it significantly reduces the cost of subsequent traversal, querying and set operations, which tend to dominate typical usage patterns. The analysis then extends to more complex manipulations, such as unions and other set operations, which are required when combining or incrementally updating data spaces. Finally, we illustrate these aspects through concrete timing results for the construction and manipulation of large-scale Earth system data hypercubes in weather forecasting and climate science, in the context of the Destination Earth initiative [48].

3.3.2 Qube Construction

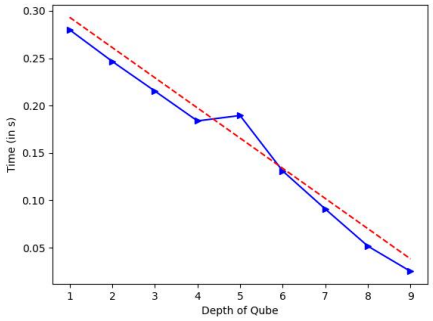
The construction time of Qubes depends primarily on the tree size and on where branching occurs in the tree. In Figure 1(a), we first study the impact of the number of leaves by constructing a flat tree that branches only at the leaves and varying their count. As expected, the construction time increases linearly with the number of leaves, since this setup requires building a linearly growing number of nodes, which is the dominant cost.
In Figure 1(b), we next examine how the branching level affects construction time by comparing trees that branch at different depths. Branching closer to the root leads to higher construction times, while branching near the leaves is faster. This linear decrease in time is expected, because earlier branching produces more nodes overall.
Finally, Figure 1(c) combines both effects by varying the number of nodes at each depth while maintaining a balanced tree. The construction time again grows linearly with the total number of nodes. However, because the number of nodes increases exponentially with the branching factor per level, even small increases in branching lead to large construction costs. Importantly, this cost is largely avoided for dense Qubes, where native compression reduces the effective number of leaves from n to 1, substantially mitigating the overhead of highly branched trees.

3.3.3 Compression

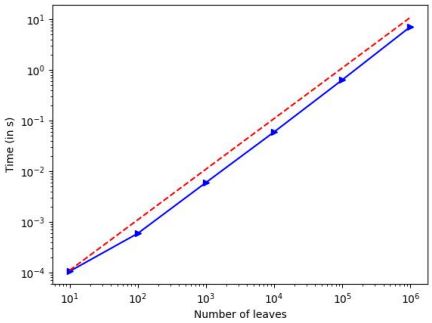
Compression is a core feature of Qubes, as it enables a highly efficient representation of the data. We therefore evaluate the cost of compressing an initially uncompressed Qube. The experimental setup is identical to the previous section, but here we measure compression time rather than construction time.
The observed trends closely mirror those for construction. In Figure 2(a), the compression cost increases linearly with the number of leaf nodes, reflecting the need to process a linearly growing number of branches. In Figure 2(b), we further observe that branching higher in the



(a) Increasing number of leaf nodes for a flat Qube.



(b) Branching at different Qube depths.



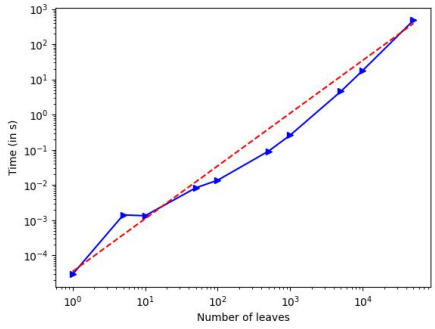
(c) Increasing number of leaves for a wide Qube.

Figure 1: Time to construct different types of Qubes. The dashed lines represent the corresponding expected linear behaviour of these timings.

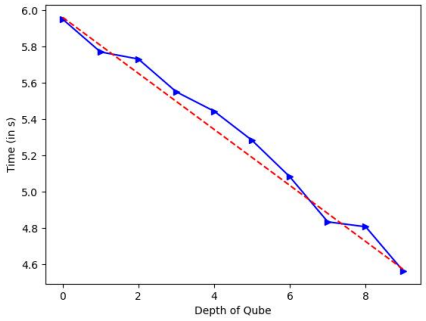
tree is less efficient for compression, since detecting and merging equivalent subtrees requires deeper recursion. Nevertheless, this cost still scales linearly and as branching moves toward the leaves, fewer nested comparisons are required, making the compression progressively cheaper.
Figure 2(c) combines these effects and again shows a linear relationship between compression time and the total number of leaves in the Qube. This behavior arises from the linear traversal of nested branches and the sequential merging of equivalent subtrees. Importantly, compression is significantly faster on already dense or partially compressed Qubes, as fewer branches need to be compared and merged.
In practice, compression is performed only once. After obtaining a maximally compressed Qube, subsequent operations benefit from this optimal representation and execute much more efficiently. Moreover, the absolute compression cost remains modest and is typically even lower in real-world settings, where Qubes tend to become denser toward the leaves, precisely where compression is most efficient as seen in Figure 2(b).

3.3.4 Set Operations

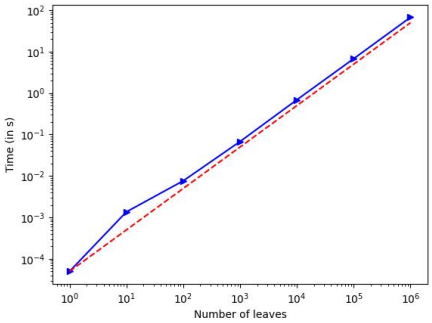
Once Qubes are constructed, the primary operations of interest are set operations. Since these operations exhibit similar performance characteristics, we focus on the union operation as a



(a) Increasing number of leaf nodes for a flat Qube.



(b) Branching at different Qube depths.



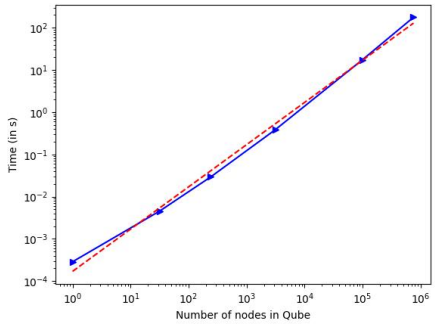
(c) Increasing number of leaves for a wide Qube with high branching factor.

Figure 2: Time to compress different types of Qubes. The dashed lines represent the corresponding expected linear behaviour of these timings.

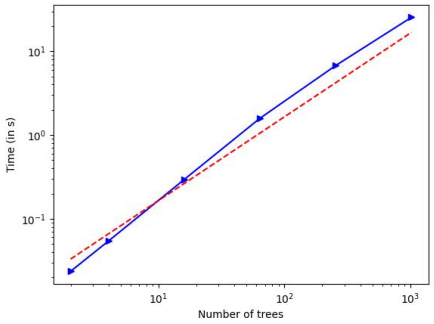
representative case. In Figure 3(a), we first compare the cost of unioning fully expanded trees with that of highly compressed Qubes. The results show that union time increases linearly with the number of nodes per tree depth, confirming the theoretical bound that set operations scale linearly with the total number of nodes in the input Qubes.
In Figure 3(b), we then study how performance scales when progressively unioning multiple Qubes of fixed size. The total cost again grows linearly, as each union is applied sequentially to the result of the previous one. This pattern is common in practice, particularly when constructing complex Qubes from several dense sub-Qubes. While the linear scaling is expected, it also suggests potential optimisations. Indeed, unioning more similar Qubes first or adopting alternative merge strategies that batch operations, could reduce the overall cost in more complex workflows.

3.3.5 Practical Considerations

In practice, constructing Qubes for real-world datasets requires combining several of the operations discussed above, including incremental construction, compression and merging. Overall performance therefore depends not only on the size of the data, but also on its structural prop-



(a) Union of two Qubes with an increasing number of total nodes in each.



(b) Union of a growing number of medium-sized Qubes.

Figure 3: Time to union Qubes together. The dashed lines represent the corresponding expected linear behaviour of these timings.

erties and on the order in which operations are applied. While naive construction and merge operations typically scale linearly with the number of entries, this quickly becomes impractical for datasets containing millions of elements, making more informed construction strategies essential.

A key optimisation strategy is to exploit structural similarity between sub-hypercubes as early as possible in the construction process. By compressing smaller Qubes before merging them into larger structures, identical or near-identical subtrees can be collapsed early. This not only reduces memory usage, but also lowers the cost of subsequent operations. Early compression also limits the growth of intermediate representations and avoids repeated work on redundant structures.

Another effective approach is to batch merge operations together so that smaller and cheaper Qubes are combined first. This delays more expensive unions on large trees until sufficient structure has already been found and compressed in intermediate results. Similarly, compression and merging can be applied to lower levels of the Qube before the resulting branches are attached to higher-level tree nodes. By only performing expensive union operations when the data structure starts to diverge, we ensure that these transformations are applied to shallower subsets of the tree, which improves the overall efficiency.

These strategies are already used in practice within the Destination Earth digital twins, where Qubes act as fast, cache-like indexes of available data to support interactive catalogue exploration. To construct these indexes, we scan the flat metadata index exposed by ECMWF's FDB data store [43, 44] and apply a combination of incremental construction, compression and merging to build a Qube that captures the structure of the available data. For the Climate digital twin [21, 7], which contains approximately 8.6 million data entries, Qube construction takes on the order of one day. In contrast, the smaller Extremes digital twin [13] can be regenerated daily in roughly one hour.

Despite these optimisations, Qube construction remains computationally expensive and should be viewed as a costly indexing step rather than a lightweight operation. Once built, however, Qubes provide a compact representation of the data structure and enable very fast selection and traversal operations. This makes them highly effective as query interfaces compared to direct access to the underlying data store. Overall, Qubes are best understood as slow-moving but fast index caches. They are expensive to generate, yet highly efficient to use and reuse within

interactive and operational workflows.

4 An Integrated Data Extraction System

The tree-based data hypercube paradigm introduced in the previous section provides a compact and expressive representation of complex, sparse and irregular data spaces. We have discussed how structural constraints and conditional data availability can be encoded directly in this data representation itself, and have shown, using Qubes as a concrete instantiation of this paradigm, how this representation can be efficiently constructed and reused.
In this section, we build on these foundations and present a complete feature extraction system that integrates structure-aware data hypercube representations directly into an operational data access pipeline. Rather than treating data hypercubes as passive data containers, the system incorporates data structure, indexing and access strategies into a single coherent workflow. This allows feature extraction algorithms to reason about data availability and layout directly from the data hypercube representation, without relying on static, externally maintained rules about the data.
This section is organised as follows. We first describe the overall system architecture and the interaction between the Polytope [30], Qubed [10] and GribJump [9] software, highlighting how structure-aware data hypercube representations are incorporated into the extraction workflow. We then analyse the performance characteristics of the system, with particular emphasis on I/O efficiency and how tree-based representations influence access patterns. Finally, we discuss the implications of this approach for user interaction, showing how it enables new, more expressive access patterns and allows users to request precisely defined scientific features from complex datasets without detailed knowledge of data layout or storage constraints.

4.1 System Architecture

The integrated feature extraction system is organised around a generic, tree-based data hypercube representation that provides a uniform and faithful view of the available data. All interactions between user requests and the underlying data stores are mediated through this data hypercube tree, which encodes the actual structure of the data space rather than an idealised dense abstraction. In our implementation, this representation is realised using Qubes. Sparsity, structural constraints and branching are encoded explicitly, allowing the system to reason directly about which combinations of coordinates exist without relying on external configuration rules.
The resulting system combines three tightly integrated components, Polytope, which provides the geometric and algorithmic core for feature extraction, Qubed, which maintains a compact, tree-based index of the available data, and GribJump, which enables fine-grained, byte-level access to the underlying storage backends. Together, these components transform feature extraction from a field- or chunk-oriented operation into a structured traversal and filtering process over a precise representation of the data space.
Qubes are constructed and maintained by the Qubed software [10]. Qubed builds this representation by scanning the flat metadata index exposed by the FDB data store [43, 44] and organising the available keys into a compressed tree structure. Each node in the tree corresponds to a pair of dimension and associated coordinates, and branches represent conditional structure within the dataset, such as dimensions that are only present for specific models, resolutions or forecast steps. Because dataset structure typically evolves slowly, the resulting Qube can be constructed as part of a pre-computation phase and cached for reuse across many extraction requests.

Feature extraction is performed by Polytope [29], which operates directly on this Qube representation. User requests, expressed in terms of scientific features such as points, trajectories, regions or timeseries, are translated by Polytope into a set of constraints over the abstract data space. Rather than constructing a dense, orthogonal view of the data hypercube by taking Cartesian products of coordinate values, Polytope traverses the Qube tree and incrementally filters it by pruning branches that are incompatible with the request. This process both ensures that the system accesses only valid combinations of coordinates in the dataset, but also that all structural constraints encoded in the data are respected throughout the extraction workflow.
The filtering operation produces a reduced Qube, which shows the exact data subsets to extract from the backend stores for a given user request. For each leaf in this resulting tree, Polytope then computes the corresponding data store indices. These are forwarded to GribJump [9], which implements the backend data access layer by providing byte-level access to data contained inside of FDB data stores. Instead of loading complete GRIB fields, or layers of the atmosphere, it retrieves only the byte ranges required to satisfy the request. This both reduces system I/O and eliminates the need for post-extraction clipping. The access model remains fully compatible with existing storage backends and file formats, as it operates directly on the underlying data representations without requiring new storage layouts.
A key architectural principle of the system is the explicit decoupling of logical data organisation from physical storage. The Qube representation captures logical structure, availability and constraints, while GribJump handles physical access to stored data. Polytope acts as the coordinating layer that translates between these two domains, ensuring that extraction logic is driven by data structure rather than by assumptions about the underlying storage layout.
Compared to existing multidimensional data frameworks such as xarray [22], Zarr [50] or RasDaMan [4, 3], which typically rely on dense arrays, predefined chunking strategies or rigid data models, this architecture generalises the notion of a datacube to encompass sparse, heterogeneous and conditionally defined data spaces. By combining tree-based data hypercubes, geometric filtering algorithms as well as byte-level data access into a single framework, the system provides a flexible foundation for efficient feature extraction across a wide range of meteorological and climate datasets.
The complete extraction workflow is illustrated in Figure 4, where a Qube index is first constructed once from the FDB metadata and cached. User-defined extraction requests are then processed by Polytope through structured traversal and filtering of this index. Finally, the resulting set of valid data indices is then passed to GribJump, which retrieves only the required bytes from the data store to assemble the final extracted feature.

4.2 System Performance

Many of the performance evaluations of the Polytope–GribJump system presented in earlier work [28] continue to apply to the new extraction pipeline. An important difference, however, is that the expensive pre-computation phase previously required to construct a datacube representation during extraction is now bypassed. Instead, the system reads an existing Qube representation, which acts as a cached view of the data space. This significantly reduces performance overhead, as the pre-computation phase is now almost instantaneous, and is particularly beneficial to enable true interactive workflows.
The introduction of the Qubed abstraction adds only minimal runtime overhead because it serves as a read-only cache queried during extraction rather than as an active processing component. As with all data access systems, however, overall performance remains closely tied to access patterns and to how well these patterns align with the underlying storage layout. When the tree structure reflects how data are physically stored, it naturally highlights access paths that group nearby data, enabling more efficient retrieval and offering guidance on optimal usage

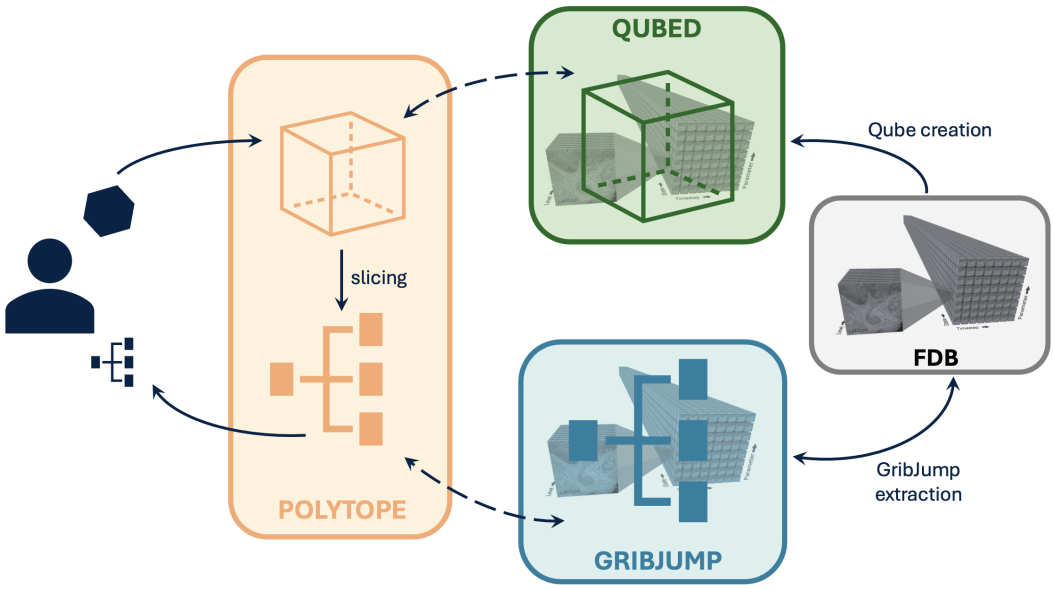


Figure 4: Full feature extraction system with the Polytope, GribJump and Qubed components.

patterns.
In practice, requests aligned with native storage layouts, such as spatial subsets within a single field, are served more quickly than requests spanning many fields. Polytope mitigates this effect by minimising total I/O and grouping access to consecutively stored data wherever possible, although algorithmic cost still increases with the size and geometric complexity of the requested feature. These effects are clearly visible in typical usage. Extracting a single full field using native access methods often completes in fractions of a second, whereas Polytope-based extraction may take a few seconds due to additional geometric processing. However, Polytope returns only a small fraction of the data, which is already advantageous in many scenarios.
As extraction scales across many fields, the benefits become more pronounced. For example, extracting a full forecast timeseries spanning 96 fields requires several seconds when using traditional full-field access, while the Polytope-based approach retrieves only a single point per field and scales minimally. When extending to ensemble forecasts involving hundreds of fields, traditional extraction can take minutes, whereas feature extraction remains on the order of seconds. This demonstrates the effectiveness of the approach for access patterns that are common in practice but poorly supported by native data layouts.
At the same time, performance remains influenced by the underlying storage system and hardware configuration. Factors such as data ordering, compression schemes, caching behaviour and storage media performance cannot be fully abstracted away. As a result, deploying the framework on a new backend or platform requires careful inspection of access patterns and dedicated performance analysis. This dependency is not unique to the proposed system and applies equally to other widely used data access frameworks, including xarray- and Zarr-based workflows. In all such systems, performance can degrade when user access patterns diverge from assumptions made during data layout and indexing.
Users should therefore be aware that extraction performance is determined not only by the feature extraction algorithm itself, but also by how data are organised and accessed at the storage

level. While the framework provides strong guarantees on minimising unnecessary data access, optimal performance ultimately requires an overall view of the full data access stack, from user request patterns down to the physical storage layer.

4.3 User-Centric Workflows and New Access Patterns

Beyond performance, a central strength of the proposed feature extraction framework lies in how it fundamentally changes the way users interact with large-scale weather and climate datasets. Traditional access mechanisms in NWP and climate data stores typically expose data at the level of complete fields or atmospheric layers. As a result, users are often forced to retrieve large volumes of data even when only a small subset is required, placing a significant burden on local storage, data transfer and post-processing before any scientific analysis can begin.

The framework developed here addresses this limitation by shifting feature extraction from a post-processing step to a core operation within the data access layer itself. The compressed tree data hypercube abstraction plays an important role in this transition. By explicitly encoding which data actually exists, along with its intrinsic constraints and branching structure, the tree view enables users to navigate and query the data space safely and naturally. Requests are guaranteed to only access existing data, as the extraction system provides strong semantic guarantees about the validity of user queries.

From a user perspective, this enables a more intuitive and application-oriented interaction model. Rather than formulating requests in terms of storage-specific concepts such as files, fields or chunks, users can express their needs directly in terms of scientific features of interest, including point time series, trajectories, regions or arbitrary spatio-temporal subsets. These high-level requests are translated internally by Polytope into precise byte-level access patterns, using the Qubed representation to reason about data availability and structure. As a result, users receive only the exact data that they need, in a format which is directly useful for downstream analysis.

This design significantly reduces the effort required to work with meteorological datasets. Indeed, users no longer need detailed knowledge of data formats, grid definitions or backend storage conventions in order to retrieve meteorological data efficiently. They can instead focus on scientific questions, whilst the extraction pipeline transparently manages the complexity of identifying, filtering and accessing the relevant data. This is particularly beneficial for non-expert users, interdisciplinary researchers and application developers, who rely on weather and climate data as inputs rather than as primary research objects.

The advantages of this approach are especially apparent in interactive and exploratory workflows. Because feature extraction retrieves only small targeted subsets of data, requests can often be served quickly enough to support iterative analysis, visualisation and prototyping. Users can explore timeseries at multiple locations, compare ensemble members or refine regions of interest without repeatedly downloading large datasets. Such interactivity is difficult to achieve with traditional full-field access and enables new classes of workflows, including web-based applications, notebooks and real-time decision-support systems.

The framework also improves accessibility by reducing local resource requirements. Full-resolution fields from modern climate simulations can easily exceed the memory and storage capacity of standard user environments. Feature-based access, by contrast, returns compact datasets that can be processed on modest local hardware or within lightweight cloud-based environments. This broadens access to large-scale weather and climate data while also supporting more sustainable data usage by minimising unnecessary data movement and duplication.

Finally, the generic nature of the feature extraction service allows it to operate across multiple data systems in a consistent manner. By separating abstract data representation from physical storage, the framework can serve as a common access layer over heterogeneous backends. This makes it well suited as a backend for standardised data access interfaces such as the OGC Envi-

ronmental Data Retrieval (EDR) API [5] and enables seamless integration with data catalogues [1], notification services [23] and downstream processing pipelines [42]. In operational contexts such as the Destination Earth digital twins [48], this unified access model allows users to discover, request and consume data through a single coherent interface, even when the underlying data are distributed across multiple systems.
Overall, the framework shifts the emphasis of data access from bulk data movement to information delivery. By allowing users to request precisely the data they need, in a form that is immediately usable, it enables more efficient, accessible and user-centred interaction with large scientific data stores. This perspective is central to the operational value of the system and highlights its potential as a foundational component for future data access services in weather, climate and Earth system science.

5 Discussion and Future Work

Feature extraction is becoming an increasingly important and widely used data access pattern in weather and climate science. As datasets continue to grow in size and complexity, users are progressively shifting away from full-field access towards more selective, application-driven queries that retrieve only the data strictly required for analysis. This trend is clearly visible in operational systems such as the Copernicus Data Store (CDS) [33], where a substantial fraction of users request spatial or spatio-temporal subsets rather than complete fields. Supporting such access patterns efficiently and robustly is therefore a key requirement for modern data services.
In this work, we have presented a complete and fully generic feature extraction framework that operates on arbitrary data hypercube representations by reasoning directly over a tree-based abstraction of the data space. By decoupling logical data organisation from physical storage and by avoiding assumptions about data completeness, layout or regularity, the framework generalises traditional datacube models to encompass sparse, heterogeneous and conditionally defined datasets. Feature extraction is no longer treated as a post-processing step applied after data retrieval, but as a core operation embedded directly into the representation and traversal of the data space.
A central strength of the proposed approach is its generality. All data- and storage-specific details are encapsulated behind generic abstractions, allowing the same extraction pipeline to be reused across different datasets, storage systems and operational contexts. This makes the framework particularly well adapted to environments in which various data sources with different formats and conventions coexist, and where user access patterns vary widely. The ability to redeploy the system with minimal changes across different platforms is a key advantage as data infrastructures continue to evolve.
The practical relevance of this approach is reinforced by observed user behaviour in operational services. In the CDS, approximately 62% of users accessing the ERA5 single-levels dataset apply an area constraint, with around 22% targeting regions smaller than $10^\circ \times 10^\circ$. These access patterns are naturally well aligned with feature extraction and stand to benefit directly from reduced data transfer and backend I/O. While the precise performance gains depend on hardware and storage configuration, the framework therefore has clear potential to alleviate I/O bottlenecks and improve scalability across the Copernicus ecosystem.
This potential is already demonstrated by the operational deployment of the Polytope-based extraction service within the Destination Earth initiative [48, 14], where it is used daily to access Weather and Climate Digital Twin data. In this setting, feature extraction acts as the primary data access mechanism, supporting interactive workflows while integrating seamlessly with existing catalogues and services. Building on this experience, the framework is planned for integration into the unified CDS, where it is expected to benefit a large fraction of users by

providing faster, more efficient and more accessible data extraction.
Several directions for future work emerge from this study. A first priority is the integration and evaluation of the framework across a broader range of backend data stores and hardware platforms. While the system is designed to minimise I/O and to align access patterns with underlying storage layouts, overall performance inevitably depends on backend-specific characteristics. Systematic evaluation across different storage technologies will therefore be essential to understand trade-offs and to guide deployment strategies.
A second direction concerns further optimisation based on observed user access patterns. By analysing how users interact with data in practice, it may be possible to adapt data hypercube representations, traversal orders or caching strategies to better match dominant workflows. This opens the door to access pattern optimisations that go beyond generic performance tuning.
Finally, the framework naturally lends itself to the integration of richer metadata into the data hypercube representation. Embedding additional information, such as grid definitions, physical data location, routing constraints or service-specific annotations, directly into the tree-based abstraction would allow the extraction system to make more informed decisions and to operate in a fully information-driven manner. Such metadata-enriched data hypercubes would further reduce the need for hard-coded assumptions and enable truly generic, self-describing data access across heterogeneous systems.
Overall, this work demonstrates how the combination of compressed tree-based data hypercube representations, geometric feature extraction and byte-level data access results in a coherent, scalable and user-centric data access framework. By bridging the gap between backend storage systems and user-driven workflows, the approach provides a practical foundation for future data services in weather, climate and Earth system science, where efficient, flexible and sustainable access to large datasets is increasingly essential.

Acknowledgements

The work presented in this paper has been produced in the context of the European Union's Destination Earth Initiative and relates to tasks entrusted by the European Union to the European Centre for Medium-Range Weather Forecasts implementing part of this Initiative with funding by the European Union. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

[1] Qubed Catalogue Browser. `https://qubed.lumi.apps.dte.destination-earth.eu/`, 2025. Accessed on 2026-02-04.

[2] Anca Anghelea, Ewelina Dobrowolska, Gunnar Brandt, Martin Reinhardt, Miguel Mahecha, Tejas Morbagal Harish, and Stephan Meissl. Deep Earth System Data Laboratory (DeepESDL). *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:13–18, 2024.

[3] Peter Baumann, Andreas Dehmel, Paula Furtado, Roland Ritsch, and Norbert Widmann. The multidimensional database system rasdaman. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 575–577, 1998.

[4] Peter Baumann, Paula Furtado, Roland Ritsch, and Norbert Widmann. The rasdaman approach to multidimensional database management. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 166–173, 1997.

[5] M Burgoyne, D Blodgett, C Heazel, and C Little. OGC API-Environmental Data Retrieval Standard. *Open Geospatial Consortium Inc., Wayland, MA, USA, OpenGIS® Implementation Specification OGC*.

[6] Deep Earth System Data Laboratory (DeepESDL) Team. Earth system data lab. `https://earthsystemdatalab.net/`, 2025. Accessed on 2025-12-30.

[7] Francisco J Doblas-Reyes, Jenni Kontkanen, Irina Sandu, Mario Acosta, Mohammed Hussam Al Turjmam, Ivan Alsina-Ferrer, Miguel Andrés-Martínez, Leo Arriola, Marvin Axness, Marc Batlle Martín, et al. The destination earth digital twin for climate change adaptation. *EGUsphere*, 2025:1–41, 2025.

[8] Brian Eaton, Jonathan Gregory, Bob Drach, Karl Taylor, Steve Hankin, John Caron, Rich Signell, Phil Bentley, Greg Rappa, Heinke Höck, et al. Netcdf climate and forecast (cf) metadata conventions, 2003.

[9] ECMWF. GribJump. `https://github.com/ecmwf/gribjump`, 2025. Accessed on 2026-02-04.

[10] European Centre for Medium-Range Weather Forecasts (ECMWF). Qubed. `https://github.com/ecmwf/qubed`, 2025. Accessed on 2026-01-05.

[11] Guillaume Eynard-Bontemps, Ryan Abernathey, Joseph Hamman, Aurelien Ponte, and Willi Rath. The PANGEO Big Data Ecosystem and its use at CNES. In *Big Data from Space (BiDS'19)…. Turning Data into insights... 19-21 février 2019, Munich, Germany*, 2019.

[12] Marzieh Fathi, Mostafa Haghi Kashani, Seyed Mahdi Jameii, and Ebrahim Mahdipour. Big data analytics in weather forecasting: A systematic review. *Archives of Computational Methods in Engineering*, 29(2):1247–1275, 2022.

[13] Estíbaliz Gascón, Irina Sandu, Benoît Vannière, Linus Magnusson, Richard Forbes, Inna Polichtchouk, Annelize Van Niekerk, Birgit Sützl, Michael Maier-Gerber, Michail Diamantakis, et al. Advances towards a better prediction of weather extremes in the destination earth initiative. Technical report, Copernicus Meetings, 2023.

[14] Thomas Geenen, Nils Wedi, Sebastian Milinski, Ioan Hadade, Balthasar Reuter, Simon Smart, James Hawkes, Emma Kuwertz, Tiago Quintino, Emanuele Danovaro, et al. Digital twins, the journey of an operational weather prediction system into the heart of destination earth. *Procedia Computer Science*, 240:99–109, 2024.

[15] Gregory Giuliani, Bruno Chatenoux, Antonio Benvenuti, Pierre Lacroix, Mattia Santoro, and Paolo Mazzetti. Monitoring land degradation at national level using satellite earth observation time-series data to support sdg15–exploring the potential of data cube. *Big Earth Data*, 4(1):3–22, 2020.

[16] Gregory Giuliani, Bruno Chatenoux, Andrea De Bono, Denisa Rodila, Jean-Philippe Richard, Karin Allenbach, Hy Dao, and Pascal Peduzzi. Building an earth observations data cube: lessons learned from the swiss data cube (sdc) on generating analysis ready data (ard). *Big Earth Data*, 1(1-2):100–117, 2017.

[17] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997.

[18] Hua-Dong Guo, Li Zhang, and Lan-Wei Zhu. Earth observation big data for climate change research. *Advances in Climate Change Research*, 6(2):108–117, 2015.

[19] Huadong Guo. Big Earth data: A new frontier in Earth and information sciences. *Big Earth Data*, 1(1-2):4–20, 2017.

[20] Venky Harinarayan, Anand Rajaraman, and Jeffrey D Ullman. Implementing data cubes efficiently. *Acm Sigmod Record*, 25(2):205–216, 1996.

[21] Jörn Hoffmann, Peter Bauer, Irina Sandu, Nils Wedi, Thomas Geenen, and Daniel Thiemert. Destination earth–a digital twin in support of climate services, 2023.

[22] Stephan Hoyer and Joe Hamman. xarray: Nd labeled arrays and datasets in python. *Journal of open research software*, 5(1):10–10, 2017.

[23] Claudio Iacopino, James N Hawkes, Tiago Quintino, and Baudouin Raoult. Aviso: Bridging hpc and cloud with high-throughput notification system for nwp data availability. In *American Meteorological Society Meeting Abstracts*, volume 101, pages 10–8, 2021.

[24] Mehmet Kaya and Reda Alhajj. Development of multidimensional academic information networks with a novel data cube based modeling method. *Information Sciences*, 265:211–224, 2014.

[25] Brian Killough. Overview of the open data cube initiative. In *IGARSS 2018-2018 IEEE international geoscience and remote sensing symposium*, pages 8629–8632. IEEE, 2018.

[26] Soohyung Kim, Hyukki Lee, and Yon Dohn Chung. Privacy-preserving data cube for electronic medical records: An experimental evaluation. *International journal of medical informatics*, 97:33–42, 2017.

[27] Tennessee Leeuwenburg, Nicholas Loveday, Elizabeth E Ebert, Harrison Cook, Mohammadreza Khanarmuei, Robert J Taggart, Nikeeth Ramanathan, Maree Carroll, Stephanie Chong, Aidan Griffiths, et al. scores: A python package for verifying and evaluating models and predictions with xarray. *arXiv preprint arXiv:2406.07817*, 2024.

[28] Mathilde Leuridan, Christopher Bradley, James Hawkes, Tiago Quintino, and Martin Schultz. Performance analysis of an efficient algorithm for feature extraction from large scale meteorological data stores. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–9, 2025.

[29] Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, Martin Schultz, and Tiago Quintino. Polytope: an algorithm for efficient feature extraction on hypercubes. *Journal of Big Data*, 12(1):1–25, 2025.

[30] Mathilde Leuridan, Adam Warde, Oisin-M, James Hawkes, James Varndell, Peter Tsrunchev, Chris Bradley, Dušan Figala, Iain Russell, Kai Kratz, Simon Smart, and Yordan Radev. ecmwf/polytope: 2.1.1, October 2025.

[31] Adam Lewis, Simon Oliver, Leo Lymburner, Ben Evans, Lesley Wyborn, Norman Mueller, Gregory Raevksi, Jeremy Hooke, Rob Woodcock, Joshua Sixsmith, et al. The australian geoscience data cube—foundations and lessons learned. *Remote Sensing of Environment*, 202:276–292, 2017.

[32] Zhenlong Li, Fei Hu, John L Schnase, Daniel Q Duffy, Tsengdar Lee, Michael K Bowen, and Chaowei Yang. A spatiotemporal indexing approach for efficient processing of big array-based climate data with mapreduce. *International Journal of Geographical Information Science*, 31(1):17–35, 2017.

[33] Angel Lopez, Carlo Buontempo, Martin Suttie, Baudouin Raoult, Edward Comyn-Platt, and James Varndell. A modernised data store infrastructure for improving the access to copernicus climate and atmosphere data and services. Technical report, Copernicus Meetings, 2022.

[34] Roxanne Suzette Lorilla. Report of new features of the sentinel hub app. *Zenodo (CERN European Organization for Nuclear Research)*, 2025.

[35] Chris Mauzey, Charles Doutriaux, Denis Nadeau, Karl E Taylor, Paul J Durack, Edward Betts, Antonio S Cofino, Piotr Florek, Emma Hogan, Jose M Rodriguez Gonzalez, et al. The climate model output rewriter (cmor). *Zenodo*, 2025.

[36] Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis. Rolap implementations of the data cube. *ACM Computing Surveys (CSUR)*, 39(4):12–es, 2007.

[37] Thomas Nicholas and Julius Busecke. Xarray-Datatree: Hierarchical Data Structures for Multi-Model Science. In *103rd AMS Annual Meeting*. AMS, 2023.

[38] Ekow J Otoo, Doron Rotem, and Sridhar Seshadri. Optimal chunking of large multidimensional arrays for data warehousing. In *Proceedings of the ACM tenth international workshop on Data warehousing and OLAP*, pages 25–32, 2007.

[39] Jonathan T Overpeck, Gerald A Meehl, Sandrine Bony, and David R Easterling. Climate data challenges in the 21st century. *science*, 331(6018):700–702, 2011.

[40] Baudouin Raoult. Mars, ecmwf's meteorological archive: Experience in managing a large archive. 2013.

[41] Russ Rew and Glenn Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[42] Iain Russell, Tiago Quintino, Baudouin Raoult, Sándor Kertész, Pedro Maciel, James Varndell, Corentin Carton de Wiart, Edward Comyn-Platt, Olivier Iffrig, James Hawkes, et al. Introducing earthkit, 2024.

[43] Simon D Smart, Tiago Quintino, and Baudouin Raoult. A scalable object store for meteorological and climate data. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–8, 2017.

[44] Simon D Smart, Tiago Quintino, and Baudouin Raoult. A high-performance distributed object-store for exascale numerical weather prediction and climate. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11, 2019.

[45] José Unpingco. *Python for probability, statistics, and machine learning*, volume 1. Springer, 2016.

[46] Wil MP Van Der Aalst. Process cubes: Slicing, dicing, rolling up and drilling down event data for process mining. In *Asia-Pacific conference on business process management*, pages 1–22. Springer, 2013.

[47] Tiffany C Vance, Thomas Huang, and Kevin A Butler. Big data in Earth science: Emerging practice and promise. *Science*, 383(6688):eadh9607, 2024.

[48] Nils Wedi, Peter Bauer, Irina Sandu, Jörn Hoffmann, Sophia Sheridan, Rafael Cereceda, Tiago Quintino, Daniel Thiemert, and Thomas Geenen. Destination earth: High-performance computing for weather and climate. *Computing in Science & Engineering*, 24(6):29–37, 2023.

[49] xarray-contrib. xarray-contrib/datatree: Prototype implementation of a tree-like hierarchical data structure for xarray. `https://github.com/xarray-contrib/datatree`, 2024. GitHub repository (archived). Accessed on 2025-12-30.

[50] Zarr Developers. Zarr: Chunked, Compressed, N-Dimensional Arrays. `https://zarr.dev/`, 2025. Accessed on 2025-12-30.

[51] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph cube: on warehousing and olap multidimensional networks. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 853–864, 2011.

8 Discussion

The previous chapters presented the design and implementation of a complete feature extraction method for efficient data access. The work mostly focused on numerical weather prediction and large-scale Earth system datasets. However, whilst these applications demonstrate the practical benefits of the approach within operational meteorological systems, the underlying concepts have broader implications beyond this specific domain.
In this chapter, we further extend the discussion to consider the wider impact of the proposed data access paradigm. In particular, we examine how the concepts introduced in this thesis relate to common problems in data-intensive scientific computing and large-scale data management more generally. The discussion therefore explores both the relevance of this work to other scientific domains that work with high-dimensional datasets, as well as its implications from a computer science perspective, including the potential for more flexible, interactive and user-driven data workflows.

8.1 A New Data Access Paradigm

The work presented in this thesis introduces a new approach to large-scale scientific data access. Instead of treating feature extraction as a post-processing step applied after data has been retrieved from storage, the proposed method performs this operation before any data is accessed by integrating feature extraction directly into the backend data system. As a result, the system is able to guarantee that only the exact subsets of data required to fulfil a given user query are read and returned. Moving feature extraction closer to the storage layer in this way not only reduces unnecessary data transfer, but it also improves the overall efficiency of the data access process.
This design represents a clear departure from more traditional data access paradigms used in scientific computing. In many existing workflows, users retrieve entire files, or large portions of them, before applying filtering or clipping operations locally. Even in more advanced database systems, such as Rasdaman [10, 11] or PostGIS [89, 125], subsetting and clipping operations are usually performed as query-time transformations after the relevant data has already been accessed from storage. Whilst these systems provide powerful abstractions for querying and manipulating multi-dimensional geospatial data, the underlying execution model still involves reading larger data regions and then applying spatial or logical filters as a subsequent processing step. Though this approach has historically been sufficient for moderately sized datasets, it becomes increasingly inefficient as datasets grow in both size and dimensionality. The approach introduced in this thesis instead shifts part of the analytical workload into the data access layer itself, allowing users to

retrieve analysis-ready subsets of data without first transferring or manipulating larger intermediate data regions.

In recent years, a range of alternative data access methods have emerged, which address similar challenges. Chunked storage formats, such as Zarr [83, 123], partition datasets into smaller and independently accessible blocks. These smaller chunks of data can then be retrieved depending on whether they intersect a user request or not. Building on this idea, more advanced tools, such as Kerchunk [24, 59], allow existing file-based formats to be virtually re-structured into chunked representations, whilst other approaches, such as Icechunk [1], extend these chunking concepts towards more dynamic and cloud-native storage abstractions. These developments have significantly improved the performance and scalability of data access on modern data stores. These approaches, however, largely rely on pre-defined chunking strategies or virtual chunk layouts, which implicitly assume certain access patterns. In highly multi-dimensional datasets, users often explore the data along many different axes, such as space, time or other metadata dimensions, and these access patterns can be difficult to anticipate in advance. As a result, even flexible chunked representations may lead to inefficient access if queries do not align well with the underlying chunk structure, sometimes requiring the retrieval of more data than is actually needed.

The feature-oriented access paradigm proposed in this thesis takes a complementary approach. Rather than optimising how data is partitioned in storage, it focuses on extracting exactly the required data elements at query time. At the same time, it operates in close interaction with the underlying storage layout and access mechanisms. In particular, the physical organisation of data, such as the contiguity of fields or the grouping of related variables, remains an important factor for performance. The goal is therefore to read as little data as necessary to satisfy a query, although the amount of data and files that must actually be read depends on how it is organised and distributed within the underlying storage system. By performing feature extraction directly within the data access layer, the system avoids the need for pre-defined storage layouts tailored to specific queries. However, this does not imply complete independence from storage structure. Instead, the approach is designed to adapt to it dynamically, leveraging favourable layouts when available whilst remaining flexible for varying access patterns. Importantly, this means that data does not need to be re-organised or re-chunked in order to support new access patterns, as the extraction logic dynamically adapts to the query itself.

Currently, implementations such as Gribjump operate directly on full GRIB fields and are therefore aligned with traditional field-based storage formats. Nevertheless, the underlying feature extraction algorithm is not inherently limited to this representation. In principle, it can be extended to operate on chunked or cloud-native datasets by providing an appropriate low-level byte access mechanism. This would enable direct extraction from formats such as Zarr, which are becoming increasingly common in the Earth sciences, and opens up opportunities to combine feature-oriented access with existing chunking strategies.

The approach presented here can therefore be seen as extending existing data access paradigms rather than replacing them. Techniques such as chunking, virtual datasets and cloud-native storage formats provide important building blocks for scalable data systems. Feature-oriented data access builds on top of these ideas

by introducing a more flexible and query-driven layer, enabling precise fine-grained access to high-dimensional datasets even when access patterns are complex or unpredictable.

8.2 Performance Considerations

Whilst the feature extraction paradigm removes many constraints imposed by storage layout, it also introduces a more nuanced performance landscape that depends on both the structure of the underlying datasets and the way in which feature extraction is performed. Importantly, performance is not completely independent of the underlying storage organisation and in practice, favourable data layouts, such as the contiguous storage of related fields, can significantly improve efficiency by reducing the number of required data accesses and enabling more effective in-memory processing. The system developed in this work follows a complex architecture, in which different execution paths are taken depending on properties such as the grid type, data representation and query geometry.

A key distinction in this context is between the cost of the feature extraction algorithm itself and the cost of accessing data from the backend storage system. These two components exhibit different scaling behaviours. The cost of data access is primarily driven by storage characteristics and access patterns, such as how data is laid out on disk, how efficiently specific byte ranges can be retrieved or how well the access aligns with underlying storage structures. In particular, efficient access seeks to minimise the number of distinct data blocks that must be loaded and to avoid repeated reads, ideally performing as much extraction as possible while data resides in memory. The cost of algorithmic feature extraction instead depends more on the properties of the datasets and the complexity of the user requests. This includes factors such as the dimension of the data, the structure of the underlying grid or the geometric properties of the request shape. This distinction is especially relevant, for example, when analysing performance across different grid types. For more structured iso-latitude grids, the original Polytope feature extraction algorithm uses efficient indexing strategies to be more performant. However, for fully unstructured grids, the performance of the Polytope algorithm can be significantly more sensitive to request sizes. In such cases, extracting larger regions or more complex geometries may require evaluating a substantial portion of the dataset. This increases the computational cost of the algorithmic feature extraction step, even if the underlying data access remains efficient.

As a result, overall system performance is determined by the interaction between these two scaling behaviours, where the first is associated with backend data access, and the second with the algorithmic performance of feature extraction. Understanding this interaction is critical for deployment across diverse environments, as the relative importance of these factors depends on the underlying storage infrastructure, data formats, and specific application requirements. This is particularly relevant in operational contexts, where performance constraints and resource utilisation must be managed carefully. The exact data storage systems, grid representations and common query patterns can all significantly influence the performance and scalability of feature extraction. In the context of this work, performance is evaluated primarily on

FDB systems, which act as field-oriented object stores and high-performance caches, often backed by memory and optimised for fast retrieval of complete meteorological fields. This has important implications for observed performance characteristics, as access to contiguous field data can be significantly faster than more fragmented access patterns typical of other disk or tape storage systems. Performance evaluations must therefore consider both the characteristics of the underlying data stores, as well as the computational cost of feature extraction for different types of datasets and applications. In practice, this requires analysing metrics such as the number of accessed fields or blocks, data volume transferred, memory reuse, and latency of individual access operations across representative hardware configurations.

Besides these core considerations, a few other additional factors influence the performance and scalability of feature extraction systems in real-world deployments. In particular, the iterative nature of the Polytope and Qubed algorithms limits the extent to which they can be parallelised. Whilst some higher-level operations, such as scanning data spaces across multiple datasets or querying independent extraction requests separately, can be parallelised efficiently, the core traversal and construction steps remain more sequential, which can limit scalability for complex queries. As a result, performance improvements are often achieved through coarser-grained parallelism, for example by distributing requests across independent regions or datasets. Furthermore, the need to operate across heterogeneous data storage systems and high-performance computing environments favours implementations in the form of flexible libraries that can be embedded directly within existing infrastructures. This enables system administrators to adapt algorithm behaviour to the characteristics of the underlying system, for instance by configuring how the data space is represented, identifying regular versus irregular dimensions, or tuning caching strategies and memory usage. The effectiveness of these choices depends strongly on the interaction between hardware characteristics, data layout and dominant query patterns, and therefore requires careful system-specific performance evaluation rather than relying on fixed default configurations.

Taken together, these observations show that, while the feature extraction paradigm enables greater flexibility in data access, its practical performance depends on how well the system is adapted to the specific properties of the data, as well as on the operational context in which it is used.

These architectural changes have important implications which extend beyond improving the efficiency of data retrieval in backend data systems however. By enabling precise and flexible data extraction directly within backend systems, the feature extraction paradigm opens the possibility for more dynamic and interactive workflows in which users can define their own access patterns rather than adapting their workflows to the structure of stored datasets. The implications of this shift become particularly visible in domains where extremely large datasets are accessed by very diverse user communities, such as operational weather forecasting for example.

8.3 User-Centric Workflows and Operational Weather Forecasting

Operational weather forecasting systems represent one of the most demanding environments for scientific data management. Modern numerical weather prediction models generate large multi-dimensional datasets at high spatial and temporal resolution. These datasets must be distributed to a wide range of users, including operational meteorological services, research institutions, downstream modelling systems, and decision-support applications. As forecasting systems increase in resolution and complexity, the volume of generated data continues to grow rapidly. At the same time, the number and diversity of users accessing these datasets is expanding. Serving these large datasets efficiently therefore represents a significant challenge for operational infrastructures.

Traditional meteorological data distribution systems often rely on file-based access patterns in which users retrieve complete model output files. In many practical use cases, however, users require only small portions of these datasets, such as specific variables, spatial regions or time ranges. For such use cases, retrieving entire files leads to unnecessary data transfer and increased I/O load on storage systems. The feature-oriented data access paradigm explored in this thesis introduces a fundamentally different interaction model for such data systems. Instead of retrieving complete files and performing subsetting operations locally, users can request exactly the subset of data required for their analysis directly from the backend system. This changes the role of the data infrastructure, and instead of dictating how users must access data, the data access system becomes a flexible service that responds directly to user-defined queries.

One of the most important consequences of this design is the natural shift toward more user-driven data workflows. In traditional data processing pipelines, workflows are often constrained by the structure of stored datasets, forcing users to adapt their downstream applications to the organisation of files or storage formats. In contrast, feature extraction allows users to define their own access patterns based on the technical requirements of their particular use cases. This enables a transition from static processing pipelines toward dynamic and interactive workflows, in which data extraction is driven by user applications rather than constrained by storage layouts. By retrieving only the required data subsets for given user queries, such as regional selections or timeseries, feature extraction avoids the expensive transfer of large unnecessary data volumes. This then directly allows for a more responsive and iterative interaction between the user and the data systems. Such capabilities are particularly important for exploratory data analysis, where researchers often need to refine their requests, test different scenarios and access multiple different subsets of data before identifying the most relevant information for their application. By lowering the cost of accessing smaller data subsets from large datasets, feature extraction enables workflows that are not only more efficient, but also inherently more interactive as it allows scientific analysis to evolve dynamically as research questions develop.

This shift toward interactive access is in particular critical in the context of digital twin ecosystems such as the Destination Earth initiative [118, 42]. These systems

are designed to provide continuous, high-resolution simulations of the Earth system, generating datasets at petabyte scales. In such environments, it is neither practical nor efficient to move complete datasets to users or downstream applications. Instead, data access must be selective, responsive and tightly integrated with analytical workflows. The deployment of the feature extraction system developed in this thesis within the Destination Earth Digital Twin Engine [42] illustrates how this can be achieved in practice. By enabling feature extraction directly within backend systems, users and services can query and retrieve only the specific data required for a given application, whether for visualisation, downstream modelling, or any other scientific analysis task. This also allows data access to be tightly coupled with other services, such as catalogue browsers or data notification systems, so that data retrieval can be triggered dynamically as part of larger workflows, rather than occurring as a separate, manual step. Within the Destination Earth initiative, for example, this new feature extraction framework integrates with services such as the Aviso data notification system [55] and STAC catalogue interfaces [34], forming part of a coordinated ecosystem for managing and accessing large-scale Earth system digital twin data. In this setting, efficient feature extraction is not just an optimisation, but a fundamental requirement for enabling scalable user-centric interaction with digital twin systems.

More broadly, these developments illustrate how feature-oriented data access can reshape the design of operational scientific data infrastructures. By supporting flexible, low-latency access to precisely defined data subsets, such systems enable new forms of interactive and user-driven workflows that are difficult to realise within traditional file-based access models. The relevance of this paradigm is not limited to meteorology, however, and similar requirements for efficient, selective data access arise across a wide range of scientific domains, as discussed in the following section.

8.4 Applications Across Scientific Domains

Although the work presented in this thesis was originally developed in the context of numerical weather forecasting, the main challenges it addresses are not unique to this domain. Across many scientific disciplines, the scale and complexity of datasets has increased significantly over the past decade as modern instruments, observational platforms and numerical simulations generate data at increasingly high spatial and temporal resolutions. As a result, researchers frequently work with datasets that are both extremely large and highly multi-dimensional. Traditional workflows often require substantial pre-processing, such as subsetting, reformatting, coordinate transformations or quality control, before meaningful analysis can begin. These pre-processing steps can represent a significant amount of the computational effort required to work with large datasets.

To address these challenges, many scientific communities have adopted the concept of analysis-ready data [6, 31], in which datasets are structured and standardised in ways that allow users to begin analysis with minimal pre-processing. This development is closely linked to the growing use of datacube architectures, which organise datasets along well-defined multi-dimensional axes such as space, time, and measured variables [13, 111]. These approaches are particularly common in Earth

observation, where satellite missions generate large observational datasets across multiple spatial, temporal and spectral dimensions. Such datasets are often organised as datacubes that allow users to analyse observations across time and space while querying specific subsets of interest. Similar data characteristics and access challenges can also be observed across a range of other scientific domains.

Remote Sensing and Earth Observation

Remote sensing and Earth observation provide one of the most natural application domains for the feature extraction approach presented in this thesis. Satellite-based systems generate large high-dimensional datacubes with dimensions spanning space, time, spectral bands and often multiple sensor platforms or processing levels [116, 5]. These datasets can therefore be highly heterogeneous, combining observational data from different instruments with varying resolutions and metadata dimensions.
In particular, many instruments provide spectral channels at different spatial resolutions, as seen in satellite systems like Sentinel-2 and Landsat, requiring the fusion of data defined on multiple grids when combining information across bands. This introduces additional complexity in data access, which can be accommodated within the proposed framework by treating each resolution level as part of a unified multi-dimensional structure and applying feature extraction consistently across them.
From a structural perspective, many Earth observation datasets are defined on regular latitude–longitude grids or projected grids, which provide a well-structured spatial representation. In these cases, the original Polytope feature extraction algorithm for structured grids can be applied directly. In addition to spatial subsetting, Earth observation workflows frequently involve filtering across multiple non-spatial dimensions, such as time ranges, spectral channels or acquisition parameters.
Moreover, cloud masking, which is one of the most common filtering operations in remote sensing, can be interpreted as introducing missing data into these high-dimensional datasets. Representing the data as a partially populated hypercube allows such masked regions to be handled naturally, contributing to the irregular and complex data spaces observed in practice.
This results in high-dimensional datacubes where access patterns are often complex and difficult to predict in advance. Feature extraction enables these multi-dimensional queries to be expressed naturally, whilst ensuring that only the required subsets are retrieved from storage. Besides, Earth observation systems increasingly rely on cloud-based and distributed data infrastructures [46, 122, 92], where minimising data transfer is critical. In this context, integrating feature extraction directly into the data access layer can significantly reduce bandwidth usage and improve responsiveness, particularly for interactive analysis and visualisation tasks.

Medical Imaging

Similar challenges arise in medical imaging, with volumetric imagery datasets for example. Imaging modalities such as magnetic resonance imaging (MRI) or computed tomography (CT) produce data defined on regular three-dimensional spatial grids, often extended with additional dimensions such as time, imaging sequences or derived physiological parameters [62, 86].

Compared to Earth observation, these datacubes are generally more compact in terms of dimensionality and most commonly consist of three spatial dimensions and time. However these can still become extremely large in high-resolution scans or longitudinal studies. The structured nature of the underlying grids allows the efficient Polytope feature extraction algorithm for regular grids to be applied directly, enabling fast identification of spatial regions of interest. In medical workflows, feature extraction is often required to isolate specific anatomical structures, extract regions for further processing or analyse temporal changes across repeated scans. These operations are usually performed after loading substantial portions of the dataset into memory. In contrast, by integrating feature extraction into the data access layer, these regions can be identified and retrieved directly, which reduces both memory usage and data transfer.
Furthermore, medical datasets can sometimes include multiple derived data products that can be viewed as additional datacube dimensions. Feature extraction provides a flexible mechanism to then navigate these datasets, supporting more targeted and efficient analysis workflows, whilst maintaining compatibility with existing imaging representations.

Astronomy

Astronomy datasets also exhibit strong structural similarities to Earth observation, although they rely on different spatial representations. Large observational surveys are commonly organised as spatial–spectral datacubes, combining angular coordinates on the celestial sphere with additional dimensions such as wavelength, time or observational parameters [30, 19].
A key characteristic of astronomy data is the use of hierarchical spherical grids such as HEALPix, which provides an equal-area tessellation of the sphere with well-defined indexing properties [23, 36, 104]. Unlike regular Cartesian grids, HEALPix introduces a structured but non-uniform spatial representation that is specifically designed for efficient analysis of spherical data. Importantly, note that the Polytope feature extraction algorithm developed in this thesis already includes support for HEALPix grids, which are iso-latitude, allowing spatial queries on the sphere to be evaluated efficiently using the existing framework. Astronomy datacubes are typically lower-dimensional than those in Earth observation, often consisting of two spatial dimensions on the sphere combined with spectral and temporal axes. However, they can still reach very large sizes due to the resolution and coverage of modern surveys. Additional metadata dimensions, such as instrument configuration or observation locations, might further increase the complexity of the data space.
Astronomical surveys also commonly contain missing or masked regions, due for example to observational constraints. This can be naturally handled within the proposed framework by representing the datacube as a partially populated data hypercube model, such as the Qube data structure. Polytope can then efficiently skip absent regions during query evaluation, ensuring that incomplete coverage does not impact performance or applicability.
In this setting, feature extraction enables efficient selection of sky regions, spectral bands or time intervals without requiring full dataset transfer. The structured nature of HEALPix grids allows the algorithm to use their hierarchical grid indexing,

ensuring that only relevant regions are processed. This makes our feature extraction approach particularly well suited for large-scale survey analysis, where users frequently perform targeted queries over specific regions of the sky.

Simulation-Based Sciences

High-dimensional simulation outputs are also common in computational physics and engineering disciplines such as computational fluid dynamics, materials science and climate modelling. Numerical simulations in these fields generate datasets defined on discretised computational meshes, which may be structured in simple cases but are increasingly unstructured, adaptive, or even dynamically evolving in time [63, 115]. In contrast to Earth system model grids, which are often based on regular 2D horizontal meshes with simple vertical layering, these meshes can be fully three-dimensional and may undergo local refinement, coarsening or deformation during the simulation. Such dynamic mesh adaptation introduce additional complexity in both spatial connectivity and temporal consistency. As a result, the associated datasets typically span three spatial dimensions and time, along with multiple physical variables such as velocity, pressure or temperature. While the overall dimensionality may be lower than in Earth observation, the complexity of the underlying grid structure introduces additional challenges for data access.
In contrast to the previous domains, unstructured meshes require the use of the more general Polytope feature extraction algorithm designed for arbitrary grid topologies. As spatial relationships cannot always be defined explicitly, the algorithm uses an additional quadtree spatial data structure, which can significantly increase computational cost, particularly when querying large regions or complex geometries. Simulation data is often defined over volume elements, such as finite volumes or finite elements [57, 40], which introduces similarities to numerical weather prediction systems, where computations are also performed over discretised spatial cells. Extending the feature extraction framework to work efficiently on such representations, however, might require adapting the unstructured grid slicer to handle three-dimensional volume elements instead of two-dimensional point-based data.
In this context, the unstructured Polytope algorithm can already be applied to irregular meshes, including three-dimensional ones. However, working with volumetric elements and complex mesh connectivity tends to increase the cost of spatial indexing and traversal, making efficient query evaluation more challenging than in the case of structured grids.

Across these different scientific domains, a wide range of strategies has been developed to address the challenges of large-scale data storage and access. Some communities rely on chunked storage layouts, multi-dimensional datacube abstractions or standardised scientific data formats, while others have developed highly specialised infrastructures tailored to the characteristics of their data. For example, particle physics experiments operate large distributed computing systems designed to process very large volumes of detector data [80, 88], whilst astronomy and Earth observation rely on domain-specific standards, indexing schemes and spatial representations such as regular grids or HEALPix. Although these approaches differ significantly in their implementation details, they all address a common underlying

challenge, which is to enable efficient and flexible access to extremely large datasets in support of complex analytical workflows.
The feature extraction paradigm introduced in this thesis provides a different perspective by focusing on extracting only the required data subsets for a given request directly from storage. Whilst the current implementation focuses on meteorological data, it already supports multiple grid representations in practice, including structured grids, reduced grids and spherical grids such as HEALPix. This demonstrates that the approach is not tied to a single spatial structure, but can operate across different grid types by leveraging structural metadata and indexing information. As a result, the same core mechanism can be applied across all of the application domains discussed above. Whether working with regular grids in Earth observation and medical imaging, spherical grids in astronomy or unstructured meshes in simulation sciences, the underlying principle, to identify and retrieve only the data required to satisfy a given query, remains the same. This positions feature-oriented access as a unifying abstraction across heterogeneous scientific data systems.

8.5 Real-Time and Operational Data Access

At the same time, deploying such a data access system for real-world applications introduces domain-specific constraints, which must be carefully considered. In operational NWP systems or digital twin environments for example, data access is often time-critical with strict requirements for forecasting and decision-making workflows. Similarly, in Earth observation platforms, interactive analysis tools depend on responsive access to large satellite datasets, whilst in medical imaging, clinical workflows may require near real-time retrieval of specific regions from high-resolution volumetric scans. In astronomy, large survey infrastructures must support efficient querying of sky regions across distributed archives, and in simulation-based sciences, data extraction is often embedded within iterative analysis and visualisation pipelines. In all of these cases, the integration of feature extraction into backend systems must account not only for performance, but also for predictability and robustness. The balance between the feature extraction cost and the cost of backend data access becomes particularly important. Whilst the proposed approach reduces unnecessary data transfer, the computational overhead of the extraction algorithm itself, especially for unstructured grids or complex query geometries, can vary depending on the dataset and use case. This leads to a dual scaling behaviour, where performance is influenced both by storage-level access patterns and by the structural properties of the underlying data.
These considerations are especially relevant when deploying the feature extraction framework across a range of heterogeneous infrastructures, including traditional file-based archives, cloud-native object stores and distributed data services. Indeed, differences in latency, throughput, as well as data layout, can significantly affect overall performance on these different systems. The feature extraction framework might then require system-level optimisations or different caching strategies tailored to specific applications. Moreover, whilst the current framework already demonstrates flexibility across multiple grid types, extending it to a broader ecosystem of data formats, such as NetCDF [95] or HDF5 [39], remains an important step.

This will require the development of specialised data access components, which implement a byte-jumping functionality similar to GribJump [32] for GRIB data files [15].
All in all, it is important to note that feature extraction is not only a conceptual shift, but also a systems-level challenge. Its successful adoption across domains such as meteorology, Earth observation, medical imaging, astronomy and simulation sciences will depend on careful integration into existing operational environments, which requires particular attention to system performance, data access patterns and diverse application-specific constraints. Addressing these aspects forms a natural direction for future work, where the focus moves from demonstrating generality toward optimising and deploying the feature extraction framework in diverse, real-world data systems.

8.6 Broader Implications

The discussion in this chapter demonstrates how feature extraction capabilities can fundamentally change the way large scientific datasets are accessed. Within the field of NWP, this data access method enables more efficient and user-centric workflows by allowing users to only retrieve precisely the data that they need for their application, without first having to transfer entire model output files or intermediate data subsets locally. As data volumes and user demands continue to grow, such fine-grained access is likely to become essential in order to maintain scalable and responsive data services.
At the same time, the characteristics that make this approach effective in meteorological systems, such as high-dimensional data, diverse user communities and increasingly interactive workflows, are not unique to this domain. Similar challenges arise in fields such as medical imaging, remote sensing, astronomy and simulation-based sciences, where large and complex datasets are routinely explored through selective queries over specific spatial, temporal or variable dimensions.
As these domains continue to move toward more interactive and exploratory modes of data analysis, the ability to retrieve precisely defined subsets directly from storage is becoming a key requirement rather than an optimisation. In this context, feature extraction provides a promising foundation for next-generation scientific data systems, supporting more flexible, responsive and user-driven workflows. More broadly, it points toward a shift in data infrastructure systems where data access frameworks are not only designed round static storage structures anymore, but also around the needs of users and applications, enabling more seamless interaction with increasingly complex data landscapes.

9 Conclusion

The increasing volume, complexity and diversity of scientific datasets is fundamentally changing how such scientific data needs to be accessed. As datasets grow in both size and dimensionality, traditional data access strategies, which are often based on retrieving complete files or large pre-defined data subsets, become increasingly inefficient and restrictive, particularly in settings where users require flexible and exploratory interaction with data.
A solution to this challenge is the introduction of feature extraction frameworks. In many existing data extraction systems however, feature extraction is usually performed only after data has already been retrieved from storage. This post-processing step then still leads to unnecessary data transfers and increased I/O costs, which can significantly impact the performance of large-scale data systems. Addressing this limitation therefore requires re-thinking how data subsetting operations are performed within the overall data extraction system.
The feature extraction method developed in this thesis introduces a fundamental design shift compared to the more traditional data extraction frameworks because it integrates feature extraction directly into the data access layer. Instead of retrieving large volumes of data and filtering them afterwards, only the exact bytes of data required for a given request is accessed in the first place. This enables more efficient data access, whilst also supporting more interactive and user-driven workflows. The following section provides a detailed summary of our work, outlining our main contributions, before finally discussing directions for future work.

9.1 Summary of Contributions

In this work, we developed a comprehensive framework for efficient feature extraction from high-dimensional branching scientific datacubes, combining algorithmic foundations with system design.
This framework is built on top of a computational geometry algorithm, which can identify data points contained within arbitrary high-dimensional polytope shapes. This provides a flexible and expressive way to define multidimensional queries, enabling users to specify complex regions of interest across spatial, temporal, as well as any other dimension. By operating directly on the available structural datacube metadata and index information, the algorithm allows precise data selection without requiring full dataset traversal.
This algorithm is then integrated within a data extraction system built on top of ECMWF's FDB meteorological data stores. This implementation shows how feature extraction can be served directly on top of operational data infrastructures, enabling efficient retrieval of meteorological features whilst significantly reducing unnecessary data transfer and backend I/O. It establishes a practical foundation for applying the

approach in large-scale production environments.
Building on this initial system, the framework is applied in different scientific contexts, including the Destination Earth digital twin environment and several Earth observation datasets. These applications demonstrate that the approach generalises beyond meteorology and can be applied to a broader class of high-dimensional datacubes. In these settings, feature extraction allows for more interactive and user-driven workflows, where data retrieval patterns are dictated by user applications rather than pre-defined storage layouts.
A key step in this generalisation is the support for multiple grid types. The framework incorporates specialised algorithms for structured grids where regularity allows for efficient query evaluation, as well as more general methods for unstructured grids, which are required in simulation-based sciences and other complex domains. Whilst unstructured grids introduce additional computational overhead, their inclusion significantly broadens the applicability of the approach to a wider range of use cases.
To unify these different representations, a generalised data hypercube abstraction is introduced, designed to capture complex, high-dimensional data spaces that may exhibit branching or conditional behaviour. In such data spaces, dimensions are not always defined uniformly and some coordinate dimensions might only exist under specific relational conditions. By integrating these complex constraints directly into the abstract data model, the framework can then support consistent and efficient feature extraction across multi-dimensional heterogeneous datasets.
Finally, these components are integrated into a complete feature extraction framework, which combines the geometric algorithm with the generalised hypercube model and backend data access. As feature extraction happens before data retrieval, the system ensures that only the necessary data subsets are accessed, reducing unnecessary data movement and improving overall efficiency.
The performance characteristics of this approach reflect a dual scaling behaviour. On the one hand, backend data access costs are reduced by limiting data retrieval to relevant subsets. On the other hand, the computational cost of the feature extraction algorithm depends both on the structure of the dataset, as well as on the complexity of the data request. For structured grid, efficient indexing minimises computational overhead, whereas in more complex scenarios with unstructured grids or highly heterogeneous data spaces, the cost of feature extraction becomes more significant and must be carefully managed.
As discussed earlier, these contributions are not limited to a single application domain, but instead define a more general paradigm for data access in large-scale scientific systems. Many scientific fields, including meteorology, Earth observation, astronomy, medical imaging and simulation-based sciences, share the challenge of working with large high-dimensional datasets and diverse evolving access patterns. In this context, feature extraction provides a flexible and scalable approach to data extraction, which complements existing storage and data processing strategies by enabling direct fine-grained extraction of relevant data.
More generally, this work shows how integrating feature extraction directly into the data access layer can support a transition towards more dynamic user-driven data workflows. By allowing users to define data requests based on their applications, rather than adapting to pre-defined data layouts, the framework enables more

custom and responsive workflows across a wide range of scientific applications.

9.2 Future Work

This thesis presents a general feature extraction framework that is applicable across a wide range of scientific data spaces. While its effectiveness has been demonstrated for meteorological and Earth system applications, the work also identifies several clear directions for future research and development. These directions fall into four main areas, which are to extend the framework to new scientific application domains, support AI- and Machine Learning (ML)–driven workflows, further develop the underlying system to improve performance, scalability and support for additional data formats and standards, and integrate the framework more tightly with surrounding data infrastructures, including distributed data stores, catalogues and interactive digital twin services. We now discuss each of these directions in turn.

Extension to new scientific application domains

A first central direction for future work is the extension of the feature extraction framework to a broader range of scientific application fields. While the current implementation already demonstrates applicability across meteorology and Earth system science, many other domains, such as medical imaging, astronomy, and simulation-based sciences, share similar challenges related to accessing and analysing large high-dimensional datasets. Transferring the framework to these domains requires adapting it to different data representations and access mechanisms. In particular, this involves enabling efficient byte-level access for a wider range of scientific file formats, as well as then integrating the Polytope feature extraction algorithms with each of the domain-specific datacube structures. Extending support to formats such as NetCDF or HDF5 would, for example, allow the framework to operate across a much broader range of scientific data systems. More generally, this direction highlights that the core idea of feature extraction is not tied to a specific application domain, but can be applied wherever large structured datasets are accessed through repeated fine-grained queries. Extending the framework in this way would therefore establish it as a broadly applicable paradigm for scientific data access.

Extension to new application domains with a focus on AI and ML

A second direction for future work is the adoption of the proposed feature extraction framework in more scientific domains, especially those driven by AI and machine learning. Such workflows usually require repeated and fine-grained access to large datasets, making efficient, user-defined data extraction particularly valuable. Extending the algorithm to operate more efficiently on Zarr-based data would be especially relevant in this context, given its increasing use in cloud-native and ML-oriented environments. This also offers the possibility of implementing hybrid approaches that combine feature extraction with chunking strategies, and could therefore provide flexible and efficient data access patterns for training, as well as inference workflows.

Feature extraction evolution from performance and scalability to standards

A third direction concerns the further development of the extraction system itself. This includes extending support for additional data formats and community standards, such as GRIB, netCDF and OGC interfaces, allowing data to be returned in forms that integrate directly with downstream applications. At the software level, future work should also focus on improving performance, in particular by reimplementing the feature extraction algorithm in a compiled language. Such improvements would reduce execution overhead and enable tighter integration with high-performance software used in operational weather forecasting, including backend data stores and numerical models such as ECMWF's IFS. In parallel, evolving the system into a fully distributed service, which can operate across multiple backend data stores would enable larger-scale deployments and more transparent access to data distributed across heterogeneous storage systems.

Integration within digital twin and modern data infrastructure ecosystems

The fourth main area for future work focuses on deeper integration of our feature extraction technique with other data infrastructure components and digital twin ecosystems. Closer coupling with data catalogues and notification services could, for example, allow dataset updates and extraction metadata to be registered automatically and used during data access. This information could be used to reduce lookup costs, simplify configuration, and further improve interactive performance. In addition, the generalised datacube abstraction introduced in this thesis provides a foundation for higher-level services, such as interactive exploration tools and query cost estimation. These capabilities would support more user-driven and adaptive data access patterns, which are particularly important in the context of large-scale Earth system digital twins.

Overall, this work lays the foundation for a new class of scalable and user-centric data access services. By enabling efficient, fine-grained access to complex and high-dimensional scientific datasets, it addresses some of the fundamental limitations of existing data extraction approaches. The framework is particularly well suited to supporting emerging AI- and machine-learning–driven applications, where flexible and repeated access to precisely defined data subsets is essential, as well as interactive digital twin workflows that require responsive and adaptive data access. In this way, the contributions of this thesis provide a basis for future data systems that can better accommodate both increasing data volumes and increasingly diverse user requirements across a range of scientific domains.

Bibliography

[1] Ryan Abernathey, Sebastián Galkin, Deepak Cherian, Joseph J Hamman, Orestis Herodotou, Brian Davis, Lindsey Nield, and Matt Iannucci. Icechunk: An open-source, transactional, cloud-native storage engine for massive-scale array data. In *105th Annual AMS Meeting 2025*, volume 105, page 456800, 2025.

[2] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D Davis, Mark Manasse, and Rina Panigrahy. Design tradeoffs for {SSD} performance. In *2008 USENIX Annual Technical Conference (USENIX ATC 08)*, 2008.

[3] Alfred V Aho and Jeffrey D Ullman. Dynamic memories with rapid random and sequential access. *IEEE Transactions on Computers*, 100(3):272–276, 1974.

[4] Anca Anghelea, Ewelina Dobrowolska, Gunnar Brandt, Martin Reinhardt, Miguel Mahecha, Tejas Morbagal Harish, and Stephan Meissl. Deep Earth System Data Laboratory (DeepESDL). *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:13–18, 2024.

[5] Marius Appel, Florian Lahn, Wouter Buytaert, and Edzer Pebesma. Open and scalable analytics of large earth observation datasets: From scenes to multidimensional arrays using scidb and gdal. *ISPRS journal of photogrammetry and remote sensing*, 138:47–56, 2018.

[6] Dani Arribas-Bel, Mark Green, Francisco Rowe, and Alex Singleton. Open data products-a framework for creating valuable analysis ready data. *Journal of Geographical Systems*, 23(4):497–514, 2021.

[7] Ana P Barros. Digital twin earth: the next-generation earth information system, 2024.

[8] Peter Bauer, Tiago Quintino, Nils Wedi, Antonio Bonanni, Marcin Chrust, Willem Deconinck, Michail Diamantakis, Peter Düben, Stephen English, Johannes Flemming, et al. The ECMWF Scalability Programme: Progress and Plans, 02/2020 2020.

[9] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

[10] Peter Baumann, Andreas Dehmel, Paula Furtado, Roland Ritsch, and Norbert Widmann. The multidimensional database system rasdaman. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 575–577, 1998.

[11] Peter Baumann, Paula Furtado, Roland Ritsch, and Norbert Widmann. The rasdaman approach to multidimensional database management. In *Proceedings of the 1997 ACM symposium on Applied computing*, pages 166–173, 1997.

[12] Peter Baumann, Paolo Mazzetti, Joachim Ungar, Roberto Barbera, Damiano Barboni, Alan Beccati, Lorenzo Bigagli, Enrico Boldrini, Riccardo Bruno, Antonio Calanducci, et al. Big data analytics for earth sciences: the earthserver approach. *International journal of digital earth*, 9(1):3–29, 2016.

[13] Peter Baumann, Dimitar Misev, Vlad Merticariu, and Bang Pham Huu. Datacubes: Towards space/time analysis-ready data. In *Service-Oriented Mapping: Changing Paradigm in Map Production and Geoinformation Management*, pages 269–299. Springer, 2018.

[14] Peter Baumann, Angelo Pio Rossi, Brennan Bell, Oliver Clements, Ben Evans, Heike Hoenig, Patrick Hogan, George Kakaletris, Panagiota Koltsida, Simone Mantovani, et al. Fostering cross-disciplinary earth science through datacube analytics. In *Earth Observation Open Science and Innovation*, pages 91–119. Springer International Publishing Cham, 2018.

[15] Jean Claude Bergès. Support of wmo binary format (bufr and grib). In *Proceedings of the Open source GIS-GRASS user conference, Trento, Italy*, volume 11, page 13, 2002.

[16] Sabpreet Bhatti, Rachid Sbiaa, Atsufumi Hirohata, Hideo Ohno, Shunsuke Fukami, and SN Piramanayagam. Spintronics based random access memory: a review. *Materials today*, 20(9):530–548, 2017.

[17] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Pangu-weather: A 3d high-resolution model for fast and accurate global weather forecast. *arXiv preprint arXiv:2211.02556*, 2022.

[18] Julian Borrill, Leonid Oliker, John Shalf, Hongzhang Shan, and Andrew Uselton. Hpc global file system performance analysis using a scientific-application derived benchmark. *Parallel Computing*, 35(6):358–373, 2009.

[19] Sébastien Bourguignon, Hervé Carfantan, Eric Slezak, and David Mary. Sparsity-based spatial-spectral restoration of muse astrophysical hyperspectral data cubes. In *2011 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–4. IEEE, 2011.

[20] Luca Brocca, Silvia Barbetta, Stefania Camici, Luca Ciabatta, Jacopo Dari, Paolo Filippucci, Christian Massari, Sara Modanesi, Angelica Tarpanelli, Bianca Bonaccorsi, et al. A digital twin of the terrestrial water cycle: a glimpse into the future through high-resolution earth observations. *Frontiers in Science*, 1:1190191, 2024.

[21] Paul G Brown. Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 963–968, 2010.

[22] M Burgoyne, D Blodgett, C Heazel, and C Little. OGC API-Environmental Data Retrieval Standard. *Open Geospatial Consortium Inc., Wayland, MA, USA, OpenGIS® Implementation Specification OGC*.

[23] Mark R Calabretta and Boudewijn F Roukema. Mapping on the healpix grid. *Monthly Notices of the Royal Astronomical Society*, 381(2):865–872, 2007.

[24] Thierry Carval, Erwan Bodere, Julien Meillon, Mathiew Woillez, Jean Francois Le Roux, Justus Magin, and Tina Odaka. Enabling simple access to a data lake both from hpc and cloud using kerchunk and intake. In *EGU General Assembly Conference Abstracts*, pages EGU–17494, 2023.

[25] Stephen Chou, Fredrik Kjolstad, and Saman Amarasinghe. Format abstraction for sparse tensor algebra compilers. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA):1–30, 2018.

[26] Rustem Dautov and Salvatore Distefano. Quantifying volume, velocity, and variety to support (big) data-intensive application development. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2843–2852. IEEE, 2017.

[27] Deep Earth System Data Laboratory (DeepESDL) Team. Earth system data lab. `https://earthsystemdatalab.net/`, 2025. Accessed on 2025-12-30.

[28] Robert Drach and John Caron. Data representation. In *Earth System Modelling-Volume 4: IO and Postprocessing*, pages 25–37. Springer, 2013.

[29] Michel Dubois, Christoph Scheurich, and Fayé Briggs. Memory access buffering in multiprocessors. *ACM SIGARCH computer architecture news*, 14(2):434–442, 1986.

[30] Geoffrey Duniam, Vyacheslav V Kitaeff, and Andreas Wicenec. Data modelling approaches to astronomical data: Mapping large spectral line data cubes to dimensional data models. *Astronomy and Computing*, 38:100539, 2022.

[31] John L Dwyer, David P Roy, Brian Sauer, Calli B Jenkerson, Hankui K Zhang, and Leo Lymburner. Analysis ready data: enabling analysis of the landsat archive. *Remote Sensing*, 10(9):1363, 2018.

[32] ECMWF. Gribjump, 2025. Accessed on 2026-01-15.

[33] Mike Espig, Wolfgang Hackbusch, Alexander Litvinenko, Hermann G Matthies, and Elmar Zander. Efficient analysis of high dimensional data in tensor formats. In *Sparse grids and applications*, pages 31–56. Springer, 2012.

[34] European Centre for Medium-Range Weather Forecasts (ECMWF). Qubed catalogue browser. `https://qubed.lumi.apps.dte.destination-earth.eu/`, 2026. Accessed on 2026-03-23.

[35] Michael R Evans, Dev Oliver, Xun Zhou, and Shashi Shekhar. Spatial big data: Case studies on volume, velocity, and variety. In *Big Data*, pages 115–138. CRC Press, 2024.

[36] P Fernique, MG Allen, T Boch, A Oberto, FX Pineau, D Durand, C Bot, L Cambrésy, S Derriere, F Genova, et al. Hierarchical progressive surveys-multi-resolution healpix data structures for astronomical images, catalogues, and 3-dimensional data cubes. *Astronomy & Astrophysics*, 578:A114, 2015.

[37] Karine R Ferreira, Gilberto R Queiroz, Lubia Vinhas, Rennan FB Marujo, Rolf EO Simoes, Michelle CA Picoli, Gilberto Camara, Ricardo Cartaxo, Vitor CF Gomes, Lorena A Santos, et al. Earth observation data cubes for brazil: Requirements, methodology and products. *Remote Sensing*, 12(24):4033, 2020.

[38] Robson Leonardo Ferreira Cordeiro, Caetano Traina, Agma Juci Machado Traina, Julio López, U Kang, and Christos Faloutsos. Clustering very large multi-dimensional datasets with mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 690–698, 2011.

[39] Mike Folk, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. An overview of the hdf5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 workshop on array databases*, pages 36–47, 2011.

[40] LS-K Fung, AD Hiebert, and Long X Nghiem. Reservoir simulation with a control-volume finite-element method. *SPE Reservoir Engineering*, 7(03):349–357, 1992.

[41] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, 30(2):170–231, 1998.

[42] Thomas Geenen, Nils Wedi, Sebastian Milinski, Ioan Hadade, Balthasar Reuter, Simon Smart, James Hawkes, Emma Kuwertz, Tiago Quintino, Emanuele Danovaro, et al. Digital twins, the journey of an operational weather prediction system into the heart of destination earth. *Procedia Computer Science*, 240:99–109, 2024.

[43] Gregory Giuliani, Bruno Chatenoux, Andrea De Bono, Denisa Rodila, Jean-Philippe Richard, Karin Allenbach, Hy Dao, and Pascal Peduzzi. Building an earth observations data cube: lessons learned from the swiss data cube (sdc) on generating analysis ready data (ard). *Big Earth Data*, 1(1-2):100–117, 2017.

[44] Gregory Giuliani, Bruno Chatenoux, Thomas Piller, Frédéric Moser, and Pierre Lacroix. Data cube on demand (dcod): Generating an earth observation data cube anywhere in the world. *International Journal of Applied Earth Observation and Geoinformation*, 87:102035, 2020.

[45] Tilmann Gneiting and Adrian E Raftery. Weather forecasting with ensemble methods. *Science*, 310(5746):248–249, 2005.

[46] Vitor CF Gomes, Gilberto R Queiroz, and Karine R Ferreira. An overview of platforms for big earth observation data management and analysis. *Remote Sensing*, 12(8):1253, 2020.

[47] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The characteristics of cloud computing. In *2010 39th International Conference on Parallel Processing Workshops*, pages 275–279. IEEE, 2010.

[48] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997.

[49] Ioan Hadade, Daniel Klocke, Jussi Enkovaara, Tuomas Lunttila, Thomas Rackow, Jan Frederik Engels, Claudia Frauen, René Redler, Jenni Kontkanen, Thomas Jung, et al. Destination earth: The climate change adaptation digital twin. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 99–110, 2025.

[50] Sam Hatfield, Aneesh Subramanian, Tim Palmer, and Peter Düben. Improving weather forecast skill through reduced-precision data assimilation. *Monthly Weather Review*, 146(1):49–62, 2018.

[51] Daniel Hein and Ralf Berger. Terrain aware image clipping for real-time aerial mapping. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:61–68, 2018.

[52] Jörn Hoffmann, Peter Bauer, Irina Sandu, Nils Wedi, Thomas Geenen, and Daniel Thiemert. Destination earth–a digital twin in support of climate services, 2023.

[53] Stephan Hoyer and Joe Hamman. xarray: Nd labeled arrays and datasets in python. *Journal of open research software*, 5(1):10–10, 2017.

[54] Fei Hu, Chaowei Yang, John L Schnase, Daniel Q Duffy, Mengchao Xu, Michael K Bowen, Tsengdar Lee, and Weiwei Song. Climatespark: An in-memory distributed computing framework for big climate data analytics. *Computers & geosciences*, 115:154–166, 2018.

[55] Claudio Iacopino, James Hawkes, Tiago Quintino, and Baudouin Raoult. Nwp data availability notifications for meteorological workflows across hpc and cloud data centres. Technical report, Copernicus Meetings, 2021.

[56] Ilias Iliadis, Linus Jordan, Mark Lantz, and Slavisa Sarafijanovic. Performance evaluation of automated tape library systems. In *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 1–8. IEEE, 2021.

[57] Geoffrey Irving, Craig Schroeder, and Ronald Fedkiw. Volume conserving finite element simulations of deformable models. *ACM Transactions on Graphics (TOG)*, 26(3):13–es, 2007.

[58] Theodore Johnson and Ethan L Miller. Benchmarking tape system performance. In *Proc. of Joint NASA/IEEE Mass Storage Systems Symposium*, 1998.

[59] Nishadh Kalladath, Masilin Gudoshava, Shruti Nath, Jason Kinyua, Fenwick Cooper, Hannah Kimani, David Koros, Christine Maswi, Zacharia Mwai, Asaminew Teshome, et al. Weather data streaming with kerchunk: Strengthening early warning systems. In *EGU General Assembly Conference Abstracts*, pages EGU25–14610, 2025.

[60] Seong Keun Kim and Mihaela Popovici. Future of dynamic random-access memory as main memory. *MRS Bulletin*, 43(5):334–339, 2018.

[61] Youngjae Kim, Aayush Gupta, Bhuvan Urgaonkar, Piotr Berman, and Anand Sivasubramaniam. Hybridstore: A cost-efficient, high-performance storage system combining ssds and hdds. In *2011 IEEE 19th annual international symposium on modelling, analysis, and simulation of computer and telecommunication systems*, pages 227–236. IEEE, 2011.

[62] Ivo Klinkert, Kamila Chughtai, Shane R Ellis, and Ron MA Heeren. Methods for full resolution data exploration and visualization for large 2d and 3d mass spectrometry imaging datasets. *International Journal of Mass Spectrometry*, 362:40–47, 2014.

[63] Asier Lacasta, Mario Morales-Hernández, Javier Murillo, and Pilar García-Navarro. An optimized gpu implementation of a 2d free surface simulation model on unstructured meshes. *Advances in engineering software*, 78:1–15, 2014.

[64] Laks VS Lakshmanan, Jian Pei, and Jiawei Han. Quotient cube: How to summarize the semantics of a data cube. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pages 778–789. Elsevier, 2002.

[65] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[66] Simon Lang, Mihai Alexe, Matthew Chantry, Jesper Dramsch, Florian Pinault, Baudouin Raoult, Mariana CA Clare, Christian Lessig, Michael Maier-Gerber, Linus Magnusson, et al. Aifs–ecmwf's data-driven forecasting system. *arXiv preprint arXiv:2406.01465*, 2024.

[67] Mark A Lantz, Simeon Furrer, Martin Petermann, Hugo Rothuizen, Stella Brach, Luzius Kronig, Ilias Iliadis, Beat Weiss, Ed R Childers, and David Pease. Magnetic tape storage technology. *ACM Transactions on Storage*, 21(1):1–70, 2025.

[68] Namgil Lee and Andrzej Cichocki. Fundamental tensor operations for large-scale data analysis using tensor network formats. *Multidimensional Systems and Signal Processing*, 29(3):921–960, 2018.

[69] Mathilde Leuridan, Christopher Bradley, James Hawkes, Tiago Quintino, and Martin Schultz. Performance analysis of an efficient algorithm for feature extraction from large scale meteorological data stores. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–9, 2025.

[70] Mathilde Leuridan, James Hawkes, and Tiago Quintino. Polytope: Feature extraction for improved access to petabyte-scale earth observation datacubes. In *Proceedings of the International Astronautical Congress (IAC)*, 2023.

[71] Mathilde Leuridan, James Hawkes, and Tiago Quintino. Polytope: Extracting features from large-scale datacubes. In *Proceedings of the International Astronautical Congress (IAC)*, 2024.

[72] Mathilde Leuridan, James Hawkes, Tiago Quintino, and Martin Schultz. An algorithm for feature extraction from large scale meteorological data stores on irregular grids. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC 2026)*, 2026. Under review.

[73] Mathilde Leuridan, James Hawkes, Tiago Quintino, and Martin Schultz. Beyond standard datacubes: Extracting features from irregular and branching earth system data, 2026.

[74] Mathilde Leuridan, James Hawkes, Simon Smart, Emanuele Danovaro, Martin Schultz, and Tiago Quintino. Polytope: an algorithm for efficient feature extraction on hypercubes. *Journal of Big Data*, 12(1):1–25, 2025.

[75] Mathilde Leuridan, Adam Warde, Oisin-M, James Hawkes, James Varndell, Peter Tsrunchev, Chris Bradley, Dušan Figala, Iain Russell, Kai Kratz, Simon Smart, and Yordan Radev. ecmwf/polytope: 2.1.2, November 2025.

[76] Martin Leutbecher and Tim N Palmer. Ensemble forecasting. *Journal of computational physics*, 227(7):3515–3539, 2008.

[77] WB Ligon and Robert B Ross. Implementation and performance of a parallel file system for high performance distributed applications. In *Proceedings of 5th IEEE International Symposium on High Performance Distributed Computing*, pages 471–480. IEEE, 1996.

[78] Yung-Ching Liu and Ming-Hui Wen. Comparison of head-up display (hud) vs. head-down display (hdd): driving performance of commercial vehicle operators in taiwan. *International Journal of Human-Computer Studies*, 61(5):679–697, 2004.

[79] Jan Mandel, Martin Vejmelka, Adam Kochanski, Angel Farguell, James Haley, Derek Mallia, and Kyle Hilburn. An interactive data-driven hpc system for forecasting weather, wildland fire, and smoke. In *2019 IEEE/ACM HPC for Urgent Decision Making (UrgentHPC)*, pages 35–44. IEEE, 2019.

[80] Manuel Martín-Márquez. Big data analytics as a service infrastructure: challenges, desired properties and solutions. In *Journal of Physics: Conference Series*, volume 664, page 042034. IOP Publishing, 2015.

[81] John M May. *Parallel I/O for high performance computing*. Morgan Kaufmann, 2001.

[82] John Michalakes. Hpc for weather forecasting. In *Parallel Algorithms in Computational Science and Engineering*, pages 297–323. Springer, 2020.

[83] Josh Moore and Susanne Kunis. Zarr: a cloud-optimized storage for interactive access of large arrays. In *Proceedings of the Conference on Research Data Infrastructure*, volume 1, 2023.

[84] Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis. Rolap implementations of the data cube. *ACM Computing Surveys (CSUR)*, 39(4):12–es, 2007.

[85] Jerome Namias. Long range weather forecasting—history, current status and outlook. *Bulletin of the American Meteorological Society*, 49(5-1):438–470, 1968.

[86] AJ Neal, G Sivewright, and R Bentley. Evaluation of a region growing algorithm for segmenting pelvic computed tomography images during radiotherapy planning. *The British Journal of Radiology*, 67(796):392–395, 1994.

[87] Dieu My T Nguyen, Johana Chazaro Cortes, Marina M Dunn, and Alexey N Shiklomanov. Impact of chunk size on read performance of zarr data in cloud-based object stores. *Authorea Preprints*, 2023.

[88] C Nieke, M Lassnig, L Menichetti, E Motesnitsalis, and D Duellmann. Analysis of cern computing infrastructure and monitoring data. In *Journal of Physics: Conference Series*, volume 664, page 052029. IOP Publishing, 2015.

[89] Regina Obe and Leo Hsu. *PostGIS in action*. Simon and Schuster, 2021.

[90] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[91] Xabier Pedruzo-Bagazgoitia, Tobias Becker, Sebastian Milinski, Thomas Rackow, Irina Sandu, Souhail Boussetta, Emanuel Dutra, Ioan Hadade, Joao

Martins, Joe McNorton, et al. Demonstrating the potential of km-scale multi-annual coupled global simulations in nextgems: a (urban) surface perspective. Technical report, Copernicus Meetings, 2024.

[92] Dana Petcu, Silviu Panica, Marian Neagul, Marc Frîncu, Daniela Zaharie, Radu Ciorba, and A Dinţs. Earth observation data processing in distributed systems. *Informatica*, 34(4), 2010.

[93] Satish Puri and Sushil K Prasad. A parallel algorithm for clipping polygons with improved bounds and a distributed overlay processing system using mpi. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 576–585. IEEE, 2015.

[94] Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: An overview. In *IEEE international conference on cloud computing*, pages 626–631. Springer, 2009.

[95] Russ Rew and Glenn Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

[96] Sanam Shahla Rizvi and Tae-Sun Chung. Flash ssd vs hdd: High performance oriented modern embedded and multimedia storage systems. In *2010 2nd international conference on computer engineering and technology*, volume 7, pages V7–297. IEEE, 2010.

[97] Matthew NO Sadiku, Sarhan M Musa, and Omonowo D Momoh. Cloud computing: opportunities and challenges. *IEEE potentials*, 33(1):34–36, 2014.

[98] Sunita Sarawagi and Michael Stonebraker. Efficient organization of large multidimensional arrays. In *Proceedings of 1994 IEEE 10th International conference on data engineering*, pages 328–336. IEEE, 1994.

[99] Hans Segura, Xabier Pedruzo-Bagazgoitia, Philipp Weiss, Sebastian K Müller, Thomas Rackow, Junhong Lee, Edgar Dolores-Tesillos, Imme Benedict, Matthias Aengenheyster, Razvan Aguridan, et al. nextgems: entering the era of kilometer-scale earth system modeling. *Geoscientific Model Development*, 18(20):7735–7761, 2025.

[100] Sugam Sharma, Udoyara Sunday Tim, and Shashi Gadia. Visual subsetting, conversion and complex query exploitation in large spatio-temporal databases. *Computers & Electrical Engineering*, 67:309–319, 2018.

[101] Amit Sheth. Transforming big data into smart data: Deriving value via harnessing volume, variety, and velocity using semantic techniques and technologies. In *2014 IEEE 30th International Conference on Data Engineering (ICDE)*, pages 2–2. IEEE Computer Society, 2014.

[102] Y Shiroishi, K Fukuda, I Tagawa, H Iwasaki, S Takenoiri, H Tanaka, H Mutoh, and N Yoshikawa. Future options for hdd storage. *IEEE Transactions on Magnetics*, 45(10):3816–3822, 2009.

[103] Adrian J Simmons and Anthony Hollingsworth. Some aspects of the improvement in skill of numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 128(580):647–677, 2002.

[104] Leo P Singer, B Parazin, Michael W Coughlin, Joshua S Bloom, Arien Crellin-Quick, Daniel A Goldstein, and Stéfan Van Der Walt. Healpix alchemy: fast all-sky geometry and image arithmetic in a relational database for multimessenger astronomy brokers. *The Astronomical Journal*, 163(5):209, 2022.

[105] Simon D Smart, Tiago Quintino, and Baudouin Raoult. A scalable object store for meteorological and climate data. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–8, 2017.

[106] Simon D Smart, Tiago Quintino, and Baudouin Raoult. A high-performance distributed object-store for exascale numerical weather prediction and climate. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, pages 1–11, 2019.

[107] Andrew Staniforth and John Thuburn. Horizontal grids for global weather and climate prediction models: a review. *Quarterly Journal of the Royal Meteorological Society*, 138(662):1–26, 2012.

[108] Jügen Steppeler, R Hess, U Schättler, and Luca Bonaventura. Review of numerical methods for nonhydrostatic weather prediction models. *Meteorology and Atmospheric Physics*, 82(1):287–301, 2003.

[109] Thomas R Stewart, Kenneth F Heideman, William R Moninger, and Patricia Reagan-Cirincione. Effects of improved information on the components of skill in weather forecasting. *Organizational behavior and human decision processes*, 53(2):107–134, 1992.

[110] Michael Stonebraker, Paul Brown, Donghui Zhang, and Jacek Becla. Scidb: A database management system for applications with complex analytics. *Computing in Science & Engineering*, 15(3):54–62, 2013.

[111] Peter Strobl, Peter Baumann, Adam Lewis, Zoltan Szantoi, Brian Killough, Matthew Purss, Max Craglia, Stefano Nativi, Alex Held, Trevor Dhu, et al. The six faces of the data cube. In *Proceedings of the 2017 conference on big data from space, Toulouse, France*, pages 28–30, 2017.

[112] Martin Sudmanns, Hannah Augustin, Lucas van der Meer, Andrea Baraldi, and Dirk Tiede. The austrian semantic eo data cube infrastructure. *Remote Sensing*, 13(23):4807, 2021.

[113] Sébastien Villaume, Manuel Fuentes, Shahram Najm, and Karen O'Regan. Grib mapping and indexing at ecwmf. In *American Meteorological Society Meeting Abstracts*, volume 102, pages J9–4, 2022.

[114] Paraskevi Vourlioti, Stylianos Kotsopoulos, Theano Mamouka, Apostolos Agrafiotis, Francisco Javier Nieto, Carlos Fernández Sánchez, Cecilia Grela Llerena, and Sergio García González. Maximizing the potential of numerical weather prediction models: lessons learned from combining high-performance computing and cloud computing. *Advances in Science and Research*, 20:1–8, 2023.

[115] Peng Wang, Tom Abel, and Ralf Kaehler. Adaptive mesh fluid simulations on gpu. *New Astronomy*, 15(7):581–589, 2010.

[116] Yi Wang, Nassim Ait Ali Braham, Zhitong Xiong, Chenying Liu, Conrad M Albrecht, and Xiao Xiang Zhu. Ssl4eo-s12: A large-scale multimodal, multitemporal dataset for self-supervised learning in earth observation [software and data sets]. *IEEE Geoscience and Remote Sensing Magazine*, 11(3):98–106, 2023.

[117] YA Wassie, MN Koeva, RM Bennett, and CHJ Lemmen. A procedure for semi-automated cadastral boundary feature extraction from high-resolution satellite imagery. *Journal of spatial science*, 63(1):75–92, 2018.

[118] Nils Wedi, Peter Bauer, Irina Sandu, Jörn Hoffmann, Sophia Sheridan, Rafael Cereceda, Tiago Quintino, Daniel Thiemert, and Thomas Geenen. Destination earth: High-performance computing for weather and climate. *Computing in Science & Engineering*, 24(6):29–37, 2023.

[119] NP Wedi, Peter Bauer, W Denoninck, Michail Diamantakis, M Hamrud, C Kuhnlein, S Malardel, Kristian Mogensen, G Mozdzynski, and PK Smolarkiewicz. The modelling infrastructure of the integrated forecasting system: Recent advances and future challenges. 2015.

[120] Qiumin Xu, Huzefa Siyamwala, Mrinmoy Ghosh, Tameesh Suri, Manu Awasthi, Zvika Guz, Anahita Shayesteh, and Vijay Balakrishnan. Performance analysis of nvme ssds and their implication on real world databases. In *Proceedings of the 8th ACM International Systems and Storage Conference*, pages 1–11, 2015.

[121] Qing Yang and Jin Ren. I-cash: Intelligently coupled array of ssd and hdd. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 278–289. IEEE, 2011.

[122] Xiaochuang Yao, Guoqing Li, Junshi Xia, Jin Ben, Qianqian Cao, Long Zhao, Yue Ma, Lianchong Zhang, and Dehai Zhu. Enabling the big earth observation data via cloud computing and dggs: Opportunities and challenges. *Remote Sensing*, 12(1):62, 2019.

[123] Zarr Developers. Zarr: Chunked, Compressed, N-Dimensional Arrays. `https://zarr.dev/`, 2025. Accessed on 2025-12-30.

[124] Björn Zehner. Interactive exploration of tensor fields in geosciences using volume rendering. *Computers & geosciences*, 32(1):73–84, 2006.

[125] Lijing Zhang and Jing Yi. Management methods of spatial data based on postgis. In *2010 Second Pacific-Asia Conference on Circuits, Communications and System*, volume 1, pages 410–413. IEEE, 2010.

[126] Jianing Zhao, Yubiao Pan, Huizhen Zhang, Mingwei Lin, Xin Luo, and Zeshui Xu. Inplacekv: in-place update scheme for ssd-based kv storage systems under update-intensive worklaods. *Cluster Computing*, 27(2):1527–1540, 2024.

[127] Jingren Zhou and Kenneth A Ross. Buffering accesses to memory-resident index structures. In *Proceedings 2003 VLDB Conference*, pages 405–416. Elsevier, 2003.