DAFAB AI

Extending Rucio for Enhanced Earth Observation Data Management

Dimitris Xenakis

30-09-24

7th Rucio Community Workshop

Outline

- **Overview of DaFab AI.**

  + Project’s vision, motivation, tasks

- **Rucio’s role in the DaFab ecosystem.**

  + What is/isn’t currently offered

- **Metadata extension roadmap**

  + Beyond planned enhancements

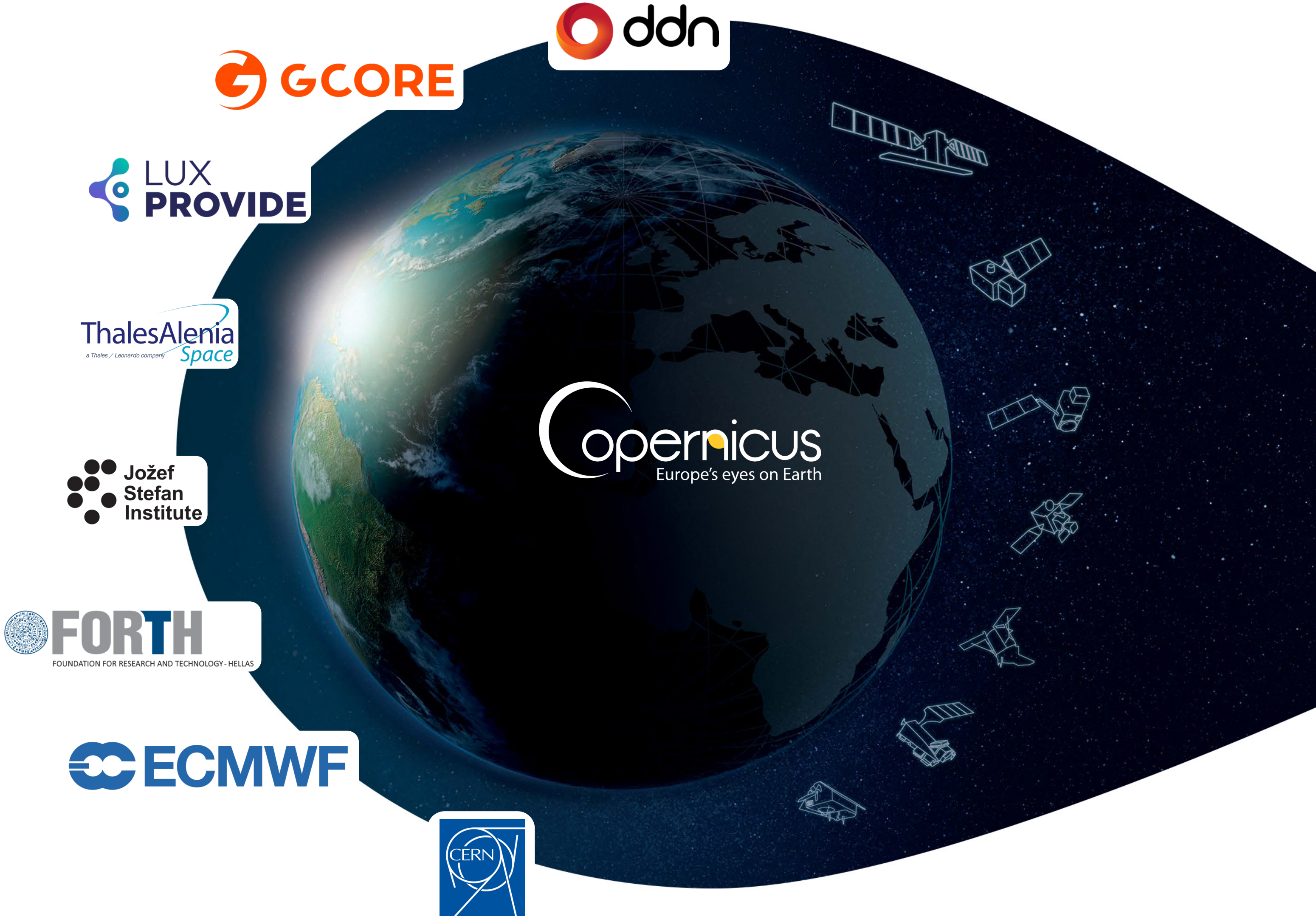
- **Key challenges**

  + Preliminary performance insights

- **Conclusion**

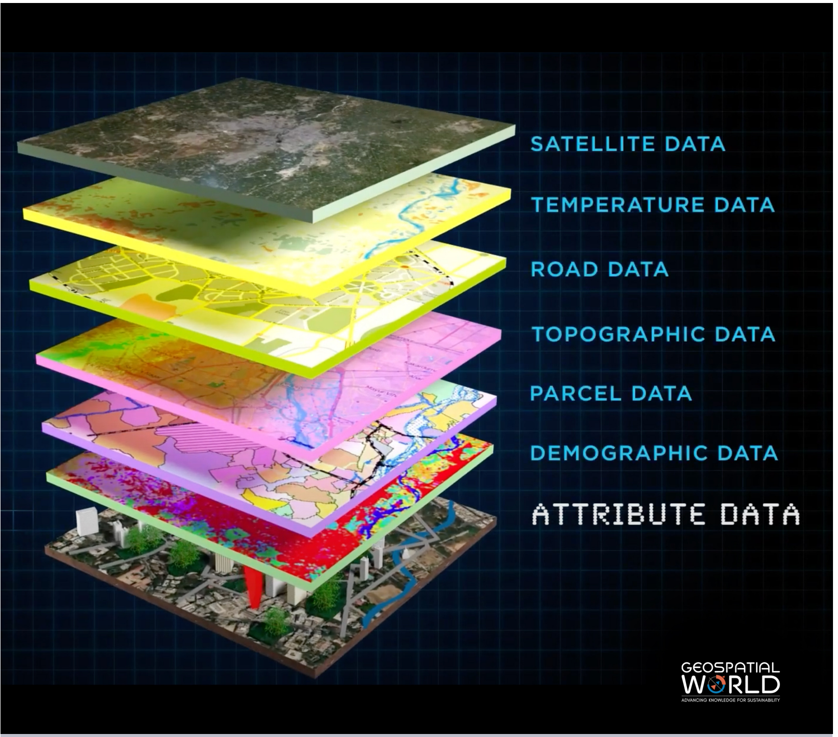
Overview of DaFab AI
Vision

Facilitate EO data exploitation through innovative cloud services and high-performance computing.



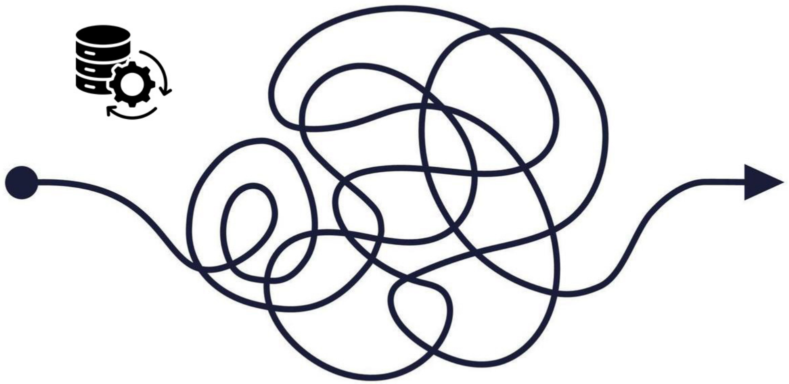
Overview of DaFab AI

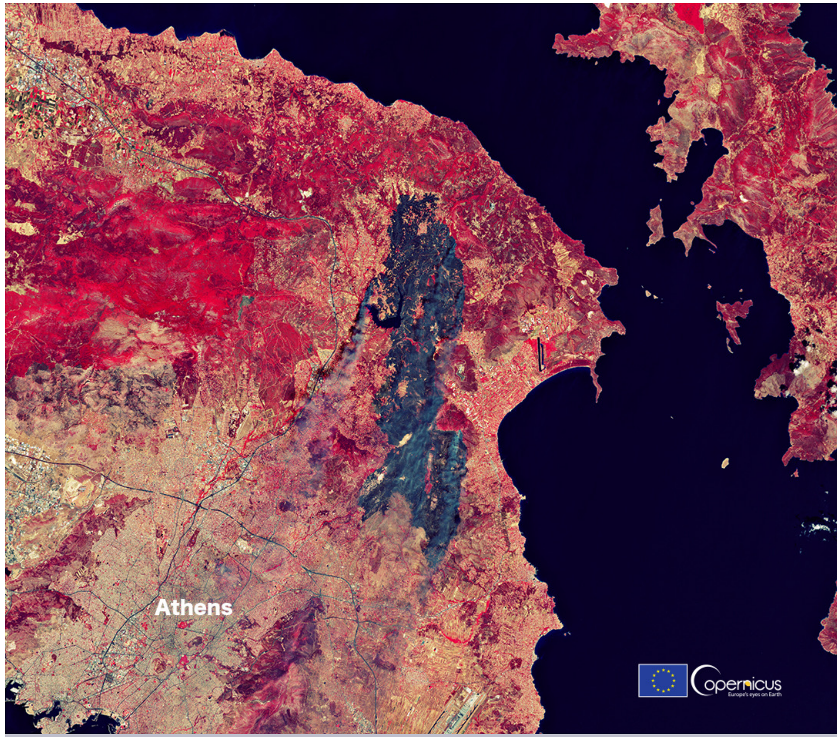
Motivation



Unified Access to Distributed and Heterogeneous Data Sources

Scalable and Sustainable Data / Workflow Management

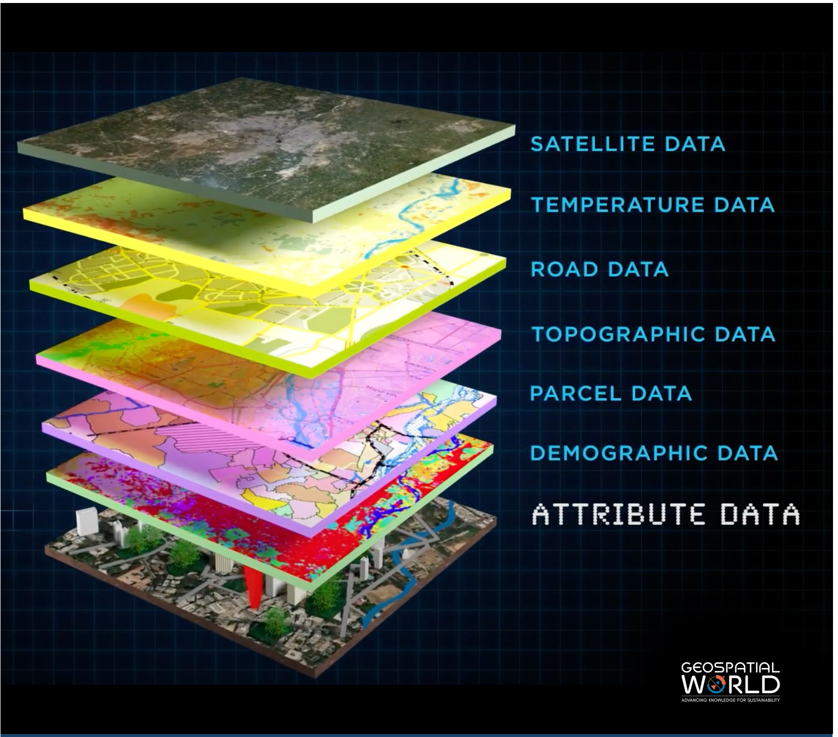




Timely EO Analysis through AI-driven Metadata Extraction

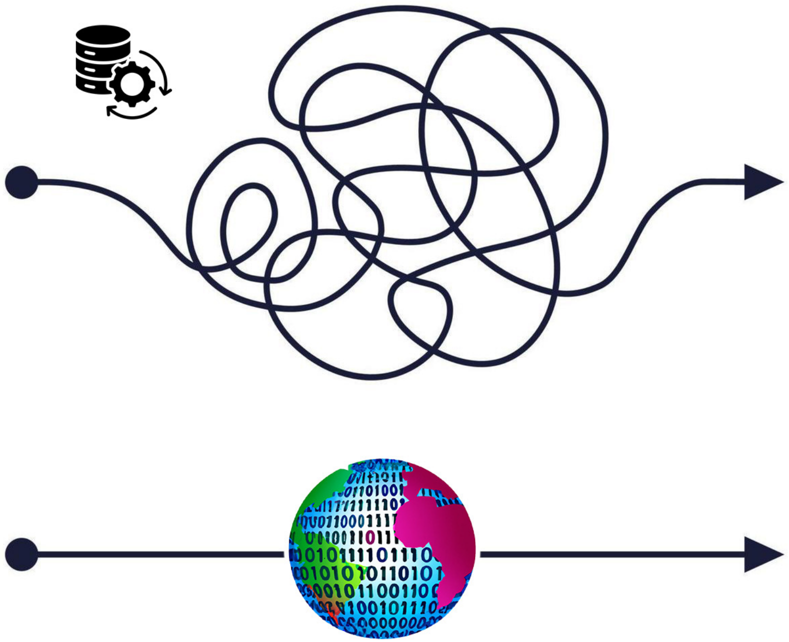
Overview of DaFab AI

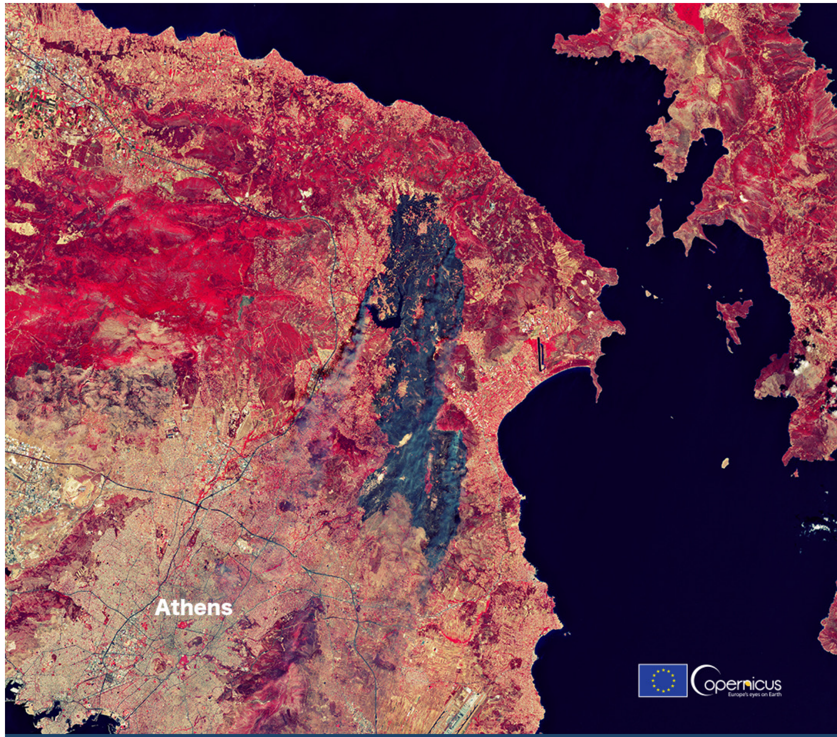
Motivation



Unified Access to Distributed and Heterogeneous Data Sources

Scalable and Sustainable Data / Workflow Management





Timely EO Analysis through AI-driven Metadata Extraction

Overview of DaFab AI
Tasks

Workgroups:

- **1:** Project Management
- **2:** AI design for metadata extraction
- **3:** Development of a Unified Metadata Catalogue
- **4:** Data Processing and Orchestration
- **5:** Use Case Implementation and Evaluation
- **6:** Communication, Dissemination, Exploitation

Overview of DaFab AI

Tasks

Workgroups:

- **1:** Project Management
- **2:** AI design for metadata extraction
- **3:** Development of a Unified Metadata Catalogue
- **4:** Data Processing and Orchestration
- **5:** Use Case Implementation and Evaluation →
- **6:** Communication, Dissemination, Exploitation

Case 1: Crop Yield Forecasting

Case 2: Flood Detection & Prediction

Overview of DaFab AI

Tasks

Workgroups:

1: Project Management

2: AI design for metadata extraction

3: Development of a Unified Metadata Catalogue →

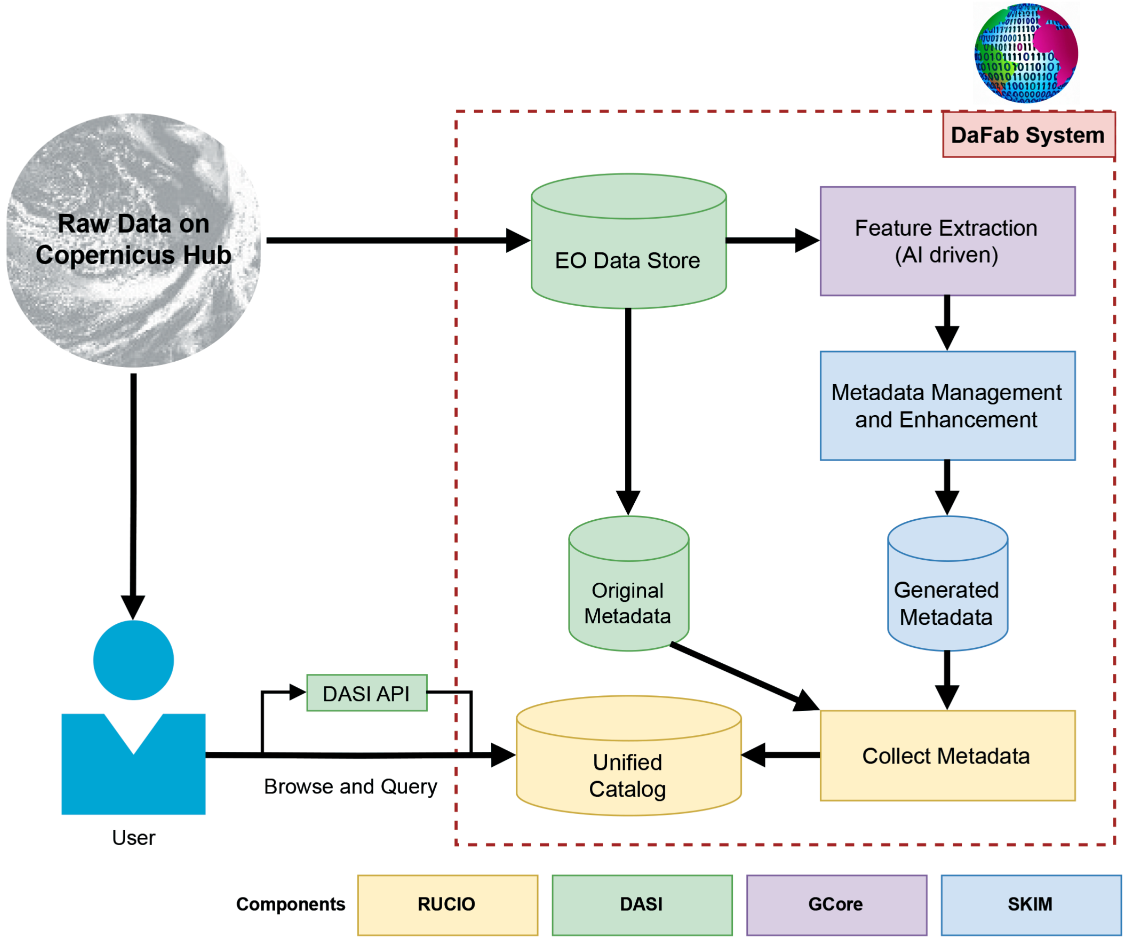
4: Data Processing and Orchestration

5: Use Case Implementation and Evaluation

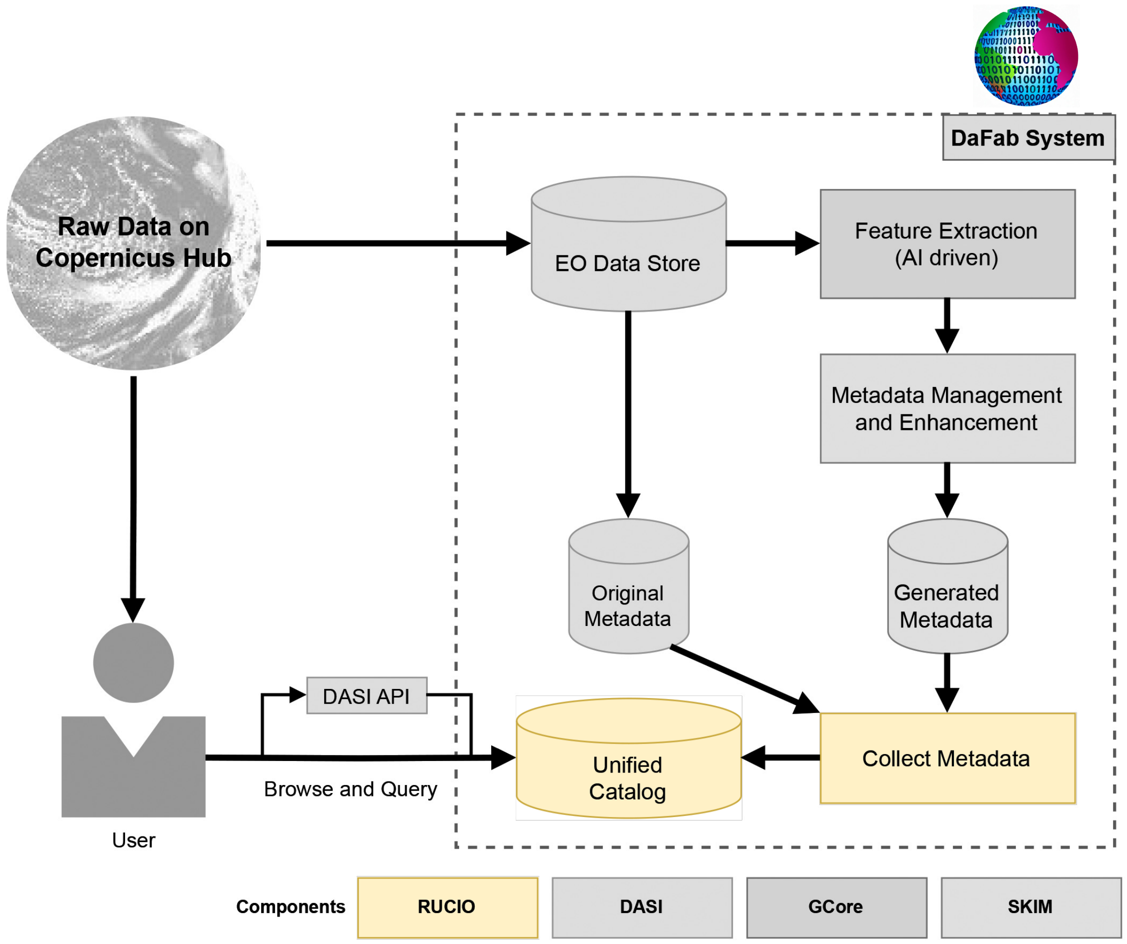
6: Communication, Dissemination, Exploitation

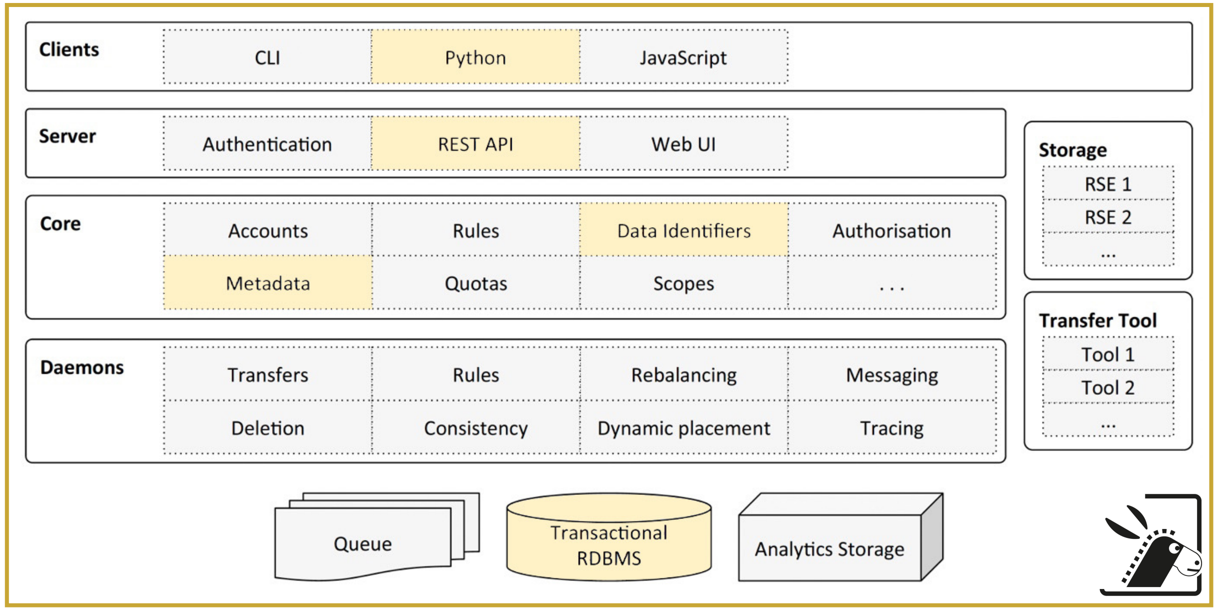
Rucio’s role in the DaFab ecosystem

High-level architecture



Rucio’s role in the DaFab ecosystem
High-level architecture





Clients	CLI	Python	JavaScript	
Server	Authentication	REST API	Web UI	
Core	Accounts	Rules	Data Identifiers	Authorisation
	Metadata	Quotas	Scopes	. . .
Daemons	Transfers	Rules	Rebalancing	Messaging
	Deletion	Consistency	Dynamic placement	Tracing

Storage	RSE 1 RSE 2 ...
Transfer Tool	Tool 1 Tool 2 ...

Rucio’s role in the DaFab ecosystem

Current metadata capabilities

A) Fixed-Columns *(default approach)*

Tables: **dids** - Contains the stored metadata in **predefined columns**

did_keys - Defines the allowed keys and their properties

did_key_map - Specifies allowed values for specific metadata keys

B) JSON Plugin *(extension)*

Tables: **did_meta** - Stores additional metadata as **Key-Value** pairs in the meta column *(JSONB / JSON / CLOB / String)*

Rucio's role in the DaFab ecosystem

Current metadata capabilities

A) Fixed-Columns *(default approach)*

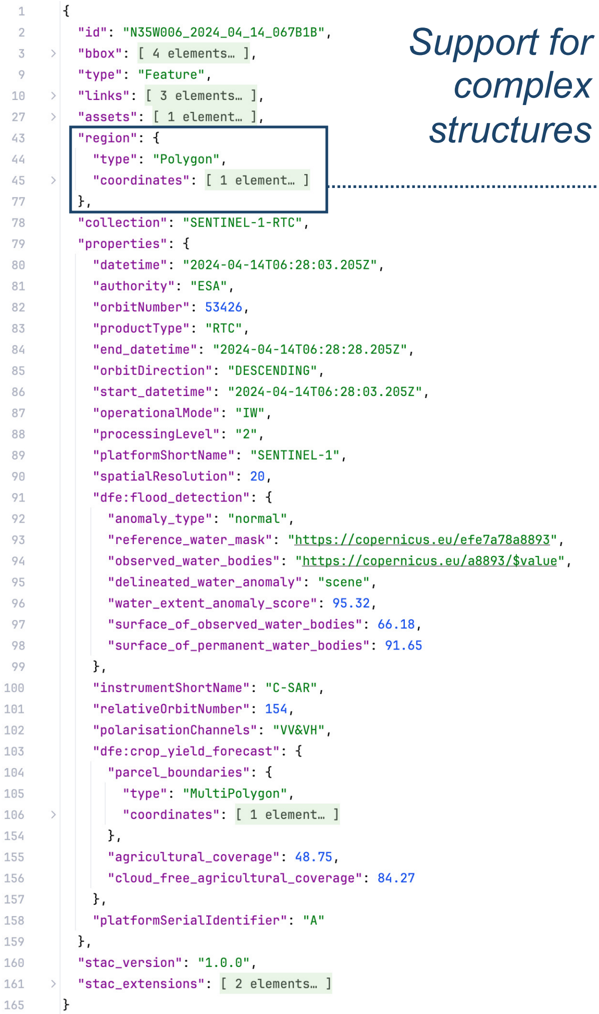
- More performant in terms of querying and indexing.
- Schema enforcement with type safety.
- Compatible with all database systems without special features.
- No overhead for storing key names repeatedly.

B) JSON Plugin *(extension)*

- Provides flexibility/extensibility for storing arbitrary metadata without altering the table's schema.

Rucio’s role in the DaFab ecosystem

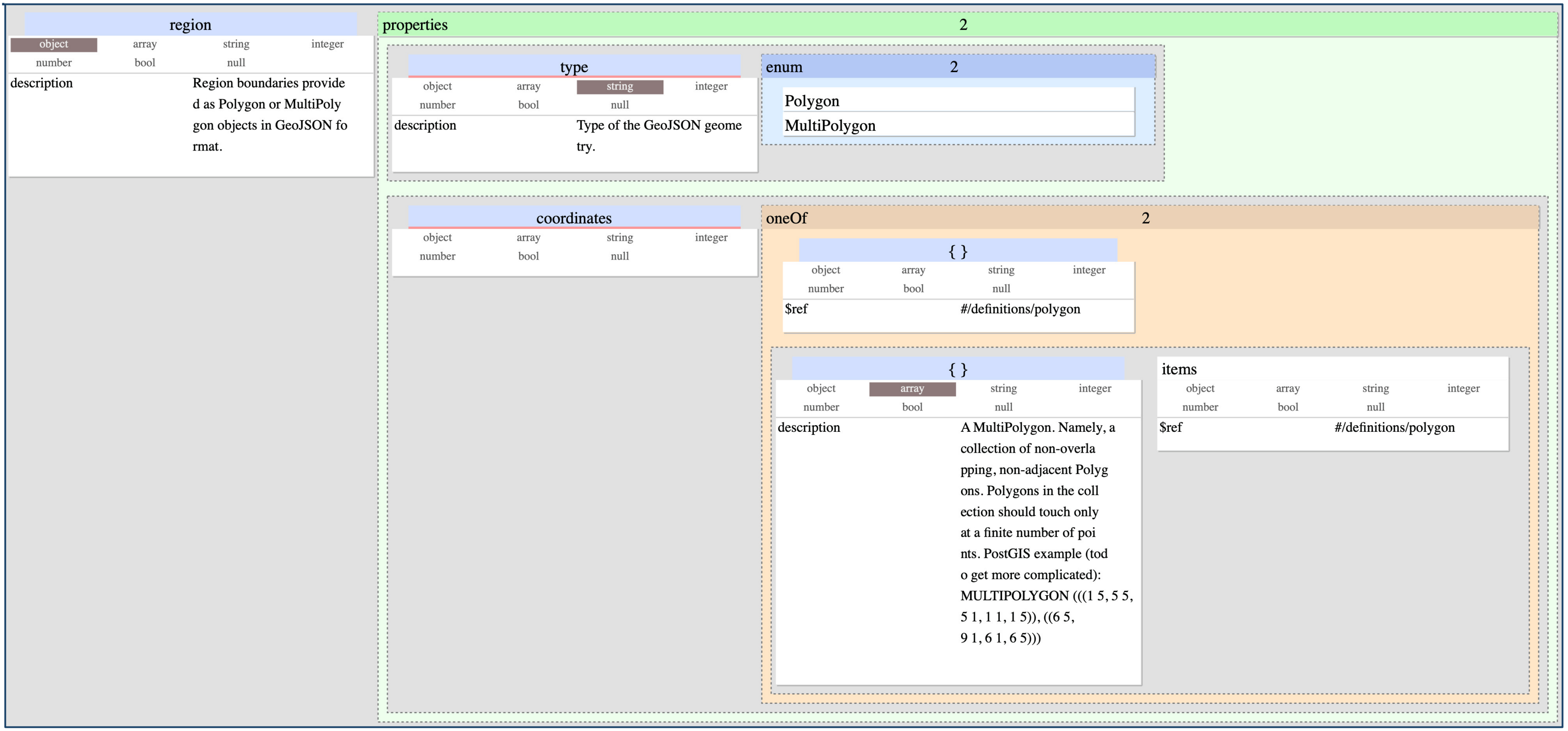
Key requirements



```
{
  "id": "N35W006_2024_04_14_067B1B",
  "bbox": [ 4 elements… ],
  "type": "Feature",
  "links": [ 3 elements… ],
  "assets": [ 1 element… ],
  "region": {
    "type": "Polygon",
    "coordinates": [ 1 element… ]
  },
  "collection": "SENTINEL-1-RTC",
  "properties": {
    "datetime": "2024-04-14T06:28:03.205Z",
    "authority": "ESA",
    "orbitNumber": 53426,
    "productType": "RTC",
    "end_datetime": "2024-04-14T06:28:28.205Z",
    "orbitDirection": "DESCENDING",
    "start_datetime": "2024-04-14T06:28:03.205Z",
    "operationalMode": "IW",
    "processingLevel": "2",
    "platformShortName": "SENTINEL-1",
    "spatialResolution": 20,
    "dfe:flood_detection": {
      "anomaly_type": "normal",
      "reference_water_mask": "https://copernicus.eu/efe7a78a8893",
      "observed_water_bodies": "https://copernicus.eu/a8893/$value",
      "delineated_water_anomaly": "scene",
      "water_extent_anomaly_score": 95.32,
      "surface_of_observed_water_bodies": 66.18,
      "surface_of_permanent_water_bodies": 91.65
    },
    "instrumentShortName": "C-SAR",
    "relativeOrbitNumber": 154,
    "polarisationChannels": "VV&VH",
    "dfe:crop_yield_forecast": {
      "parcel_boundaries": {
        "type": "MultiPolygon",
        "coordinates": [ 1 element… ]
      },
      "agricultural_coverage": 48.75,
      "cloud_free_agricultural_coverage": 84.27
    },
    "platformSerialIdentifier": "A"
  },
  "stac_version": "1.0.0",
  "stac_extensions": [ 2 elements… ]
}
```

Support for complex structures

Schema enforcement



region

object array string integer number bool null

description: Region boundaries provided as Polygon or MultiPolygon objects in GeoJSON format.

properties 2

type

object array string integer number bool null

description: Type of the GeoJSON geometry.

enum 2

Polygon

MultiPolygon

coordinates

object array string integer number bool null

oneOf 2

{ }

object array string integer number bool null

$ref #/definitions/polygon

{ }

object array string integer number bool null

description: A MultiPolygon. Namely, a collection of non-overlapping, non-adjacent Polygons. Polygons in the collection should touch only at a finite number of points. PostGIS example (todo get more complicated): MULTIPOLYGON (((1 5, 5 5, 5 1, 1 1, 1 5)), ((6 5, 9 1, 6 1, 6 5)))

items

object array string integer number bool null

$ref #/definitions/polygon

Metadata extension roadmap

Current milestones for the “extended metadata” approach

- Develop a JSONB-based metadata cataloging mechanism that satisfies DaFab’s requirements.

- Design REST API methods for..

Metadata Access of selected DID/s	DID Search with Metadata Filtering	Metadata Management
• Fetch specified/all fields	• Comparisons: ==, !=, >, <, >=, <= • Combinations: AND, OR	• Add/remove/modify fields • Edit/validate schema

- Bring the functionality to Rucio Client.

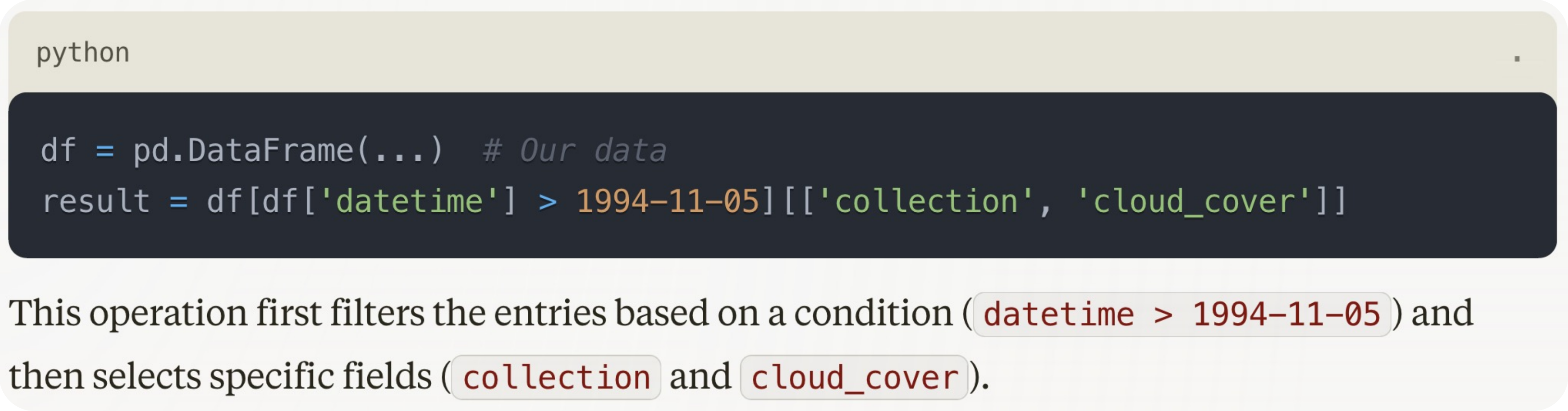
- Migrate Rucio Server from *Fixed-Column* to *JSON* while maintaining performance.

Metadata extension roadmap
and Beyond.. (open to discussion)

- More powerful API functionalities.

..e.g. *Select which JSON fields to be returned during a “DID Search with Metadata Filtering”.*

Similar to Pandas filtering in python



python

```python
df = pd.DataFrame(...)  # Our data
result = df[df['datetime'] > 1994-11-05][['collection', 'cloud_cover']]
```

This operation first filters the entries based on a condition (`datetime > 1994-11-05`) and then selects specific fields (`collection` and `cloud_cover`).

- Generalize the metadata functionality to be applicable to more queries.

Key challenges

Require addressing

Schema Evolution: *Ensure backward compatibility while allowing for future metadata schema changes.*

Query Performance: *Optimize queries/indices on JSONB data to maintain (or improve) current performance.*

Scalability: *Ensure the new system can handle increasing volumes of data and users.*

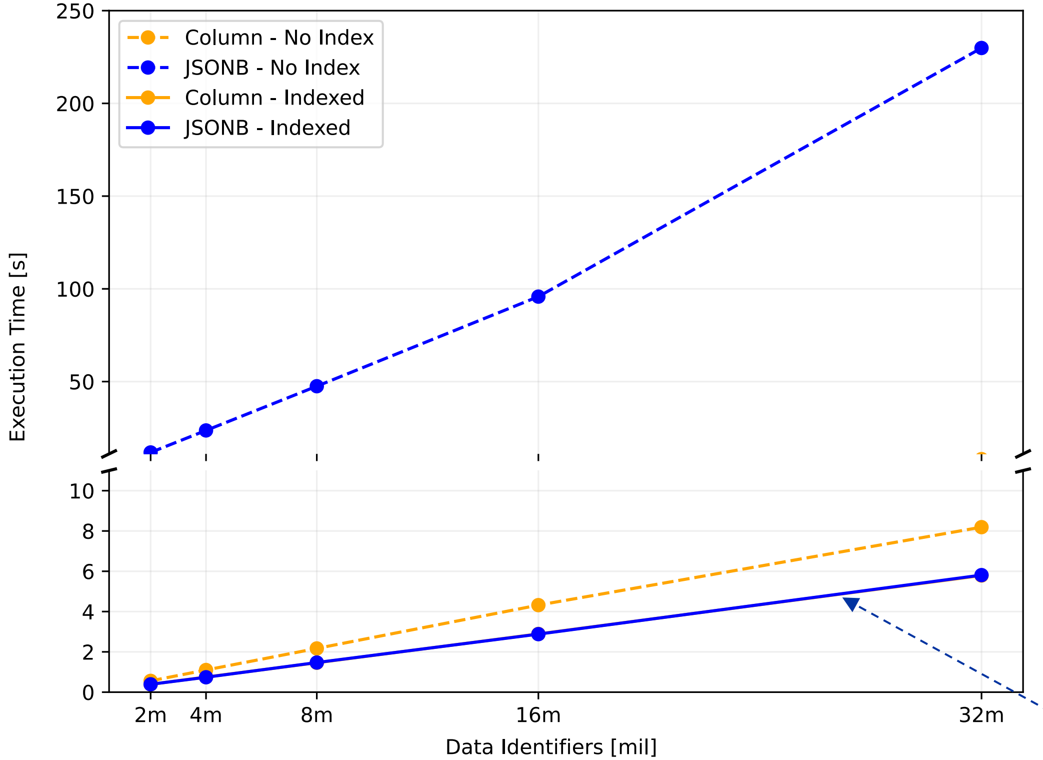
Data Migration: *Safely transition existing fixed-column metadata to the new JSON-based structure.*

Key challenges

Performance insights: Fixed-column vs JSONB

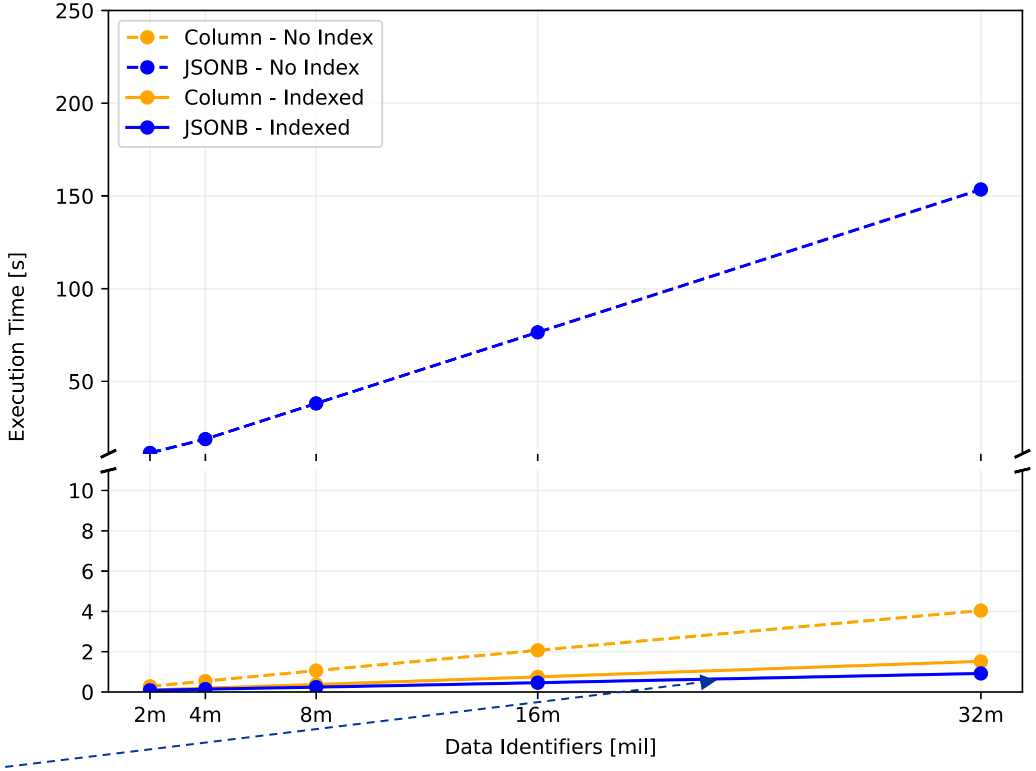
(collection = ‘S2’)

57% of table matched



(t >= '2023-01-01 01:00:00'::timestamp **AND** t <= '2024-01-01 01:00:00'::timestamp)
AND
(cloud_cover > 60.0 **AND** collection = 'S2’)

2% of table matched

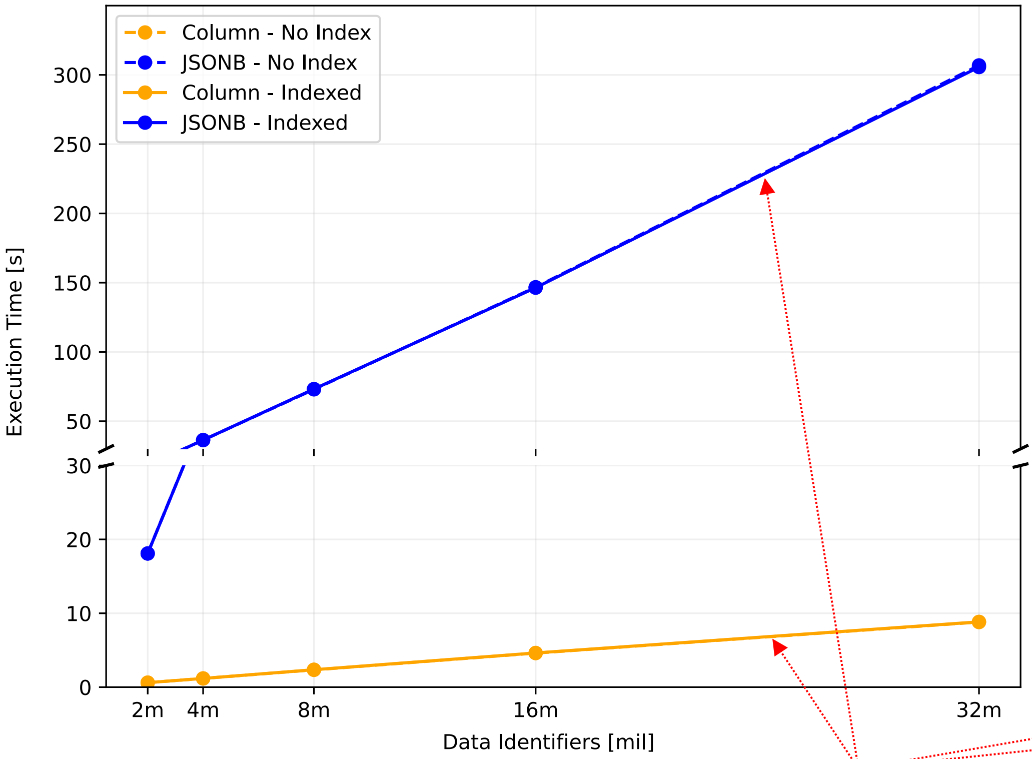


- Similar performance via indexing

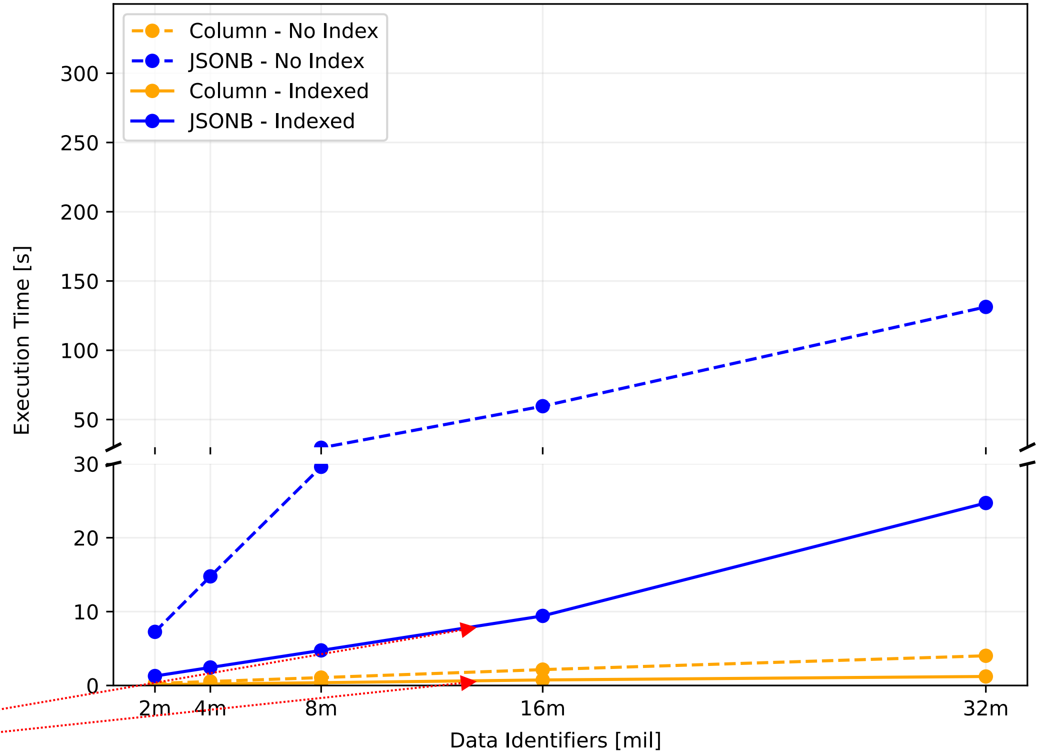
Key challenges

Performance insights: Fixed-column vs JSONB

(cloud_cover > 60.0 **OR** collection = 'S2') *74% of table matched*



(t >= '2023-01-01 01:00:00'::timestamp **AND** t <= '2024-01-01 01:00:00'::timestamp)
AND
(cloud_cover > 60.0 **OR** collection = 'S2') *5% of table matched*



- In both ***Fixed-column*** & ***JSONB*** cases, the planner avoided **any** index use for *(cloud_cover > 60.0* ***OR*** *collection = 'S2')*

- Due to the specific ***cloud_cover*** / ***collection*** statistics.
- Yet how representative would such a total indexing-rejection be?

Conclusion

- **DaFab AI goal:**
  Enhance EO data management using upcoming Rucio’s extended metadata capabilities.

- **Key developments:**
  new JSON-based DB mechanism, improved API and client, and migration from fixed-columns.

- **Challenges:**
  Schema evolution, performance, scalability, data migration.

- **Performance:**
  When indices are utilized, JSONB queries can match fixed-column performance.

Thank you for your attention

Dimitris Xenakis: d.xenakis@cern.ch

DaFab AI: www.dafab-ai.eu

Thanks for asking
What about Elasticsearch?

	JSONB	Elasticsearch
Query Complexity	Our queries are primarily **simple lookups** and **basic filtering**.	We **require advanced full-text search** and **complex nested queries**.
Scalability Concerns	Our DB growth is manageable. A **vertical scaling suffices**.	We **need horizontal scalability** that Elasticsearch provides.
Operational Overhead	We prefer maintaining a **single database** system.	We have the resources for **additional Elasticsearch clusters** (or ELK stacks).
Data Consistency	**Strong consistency is crucial** for our use case.	We can **tolerate eventual consistency** for metadata queries.
Integration Effort	**Minimal changes required** to existing Rucio codebase.	We're **prepared for significant changes** to Rucio's data access layer.
Storage Strategy	Keeping all data in **PostgreSQL simplifies our data management**.	We're **open to a hybrid storage approach** for improved query performance.
Flexibility Needs	Our metadata **schema is relatively stable**.	We need high flexibility for **frequently changing metadata structures**.