Cloud storage capable to select events upon user request for medium-sized astrophysical experiments

Alexander Kryukov\textsuperscript{1}, Igor Bychkov\textsuperscript{2}, Elena Korosteleva\textsuperscript{1}, Andrey Mikhakov\textsuperscript{2}, Min-Duc Nguyen\textsuperscript{1}

\textsuperscript{1}M.V.Lomonosov Moscow State University, D.V.Skobeltsyn Institute of Nuclear Physics, Russia
\textsuperscript{2}Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences, Russia
E-mail: kryukov@theory.sinp.msu.ru

Abstract. We present a cloud storage for collaborations of small and medium-sized experiments in the field of particle astrophysics. This storage provides a unified interface for accessing data from different experiments. In this article, we will focus on the capabilities of the storage for processing user requests for data, as well as on some technical details of the implementation of data selection. We have deployed a working prototype of the storage. Currently the prototype integrates data from such astrophysical experiments as TAIGA and KASCADE. As a result, users of the respective collaborations have the opportunity to collect scientific data from different experiments seamlessly within one request to conduct joint data analysis.

1. Introduction
Modern particle astrophysics is one of the most dynamically developing areas of modern science. It includes several scientific directions, in each of which several large experimental installations have been put into operation (see, for example, \cite{1, 2}). For such large experiments, special software has been created that provide experimental data to their participants. However, the complexity of this software makes it difficult to use for small and medium-sized collaborations. At the same time, there is a lack of suitable user-friendly software for such small groups. We have developed a distributed data storage \cite{3} to provide unified access to local storages of multiple small and medium-sized experiments, making it easier to conduct joint (multi-messenger) data analysis. We named this storage AstroDS. To test the AstroDS functionality and performance, we use the data of two experiments: TAIGA \cite{4, 5} and KASCADE \cite{6, 7}. The development of the AstroDS is carried out within the framework of the Russian-German initiative \cite{8}.

In this article, we will focus on the capabilities of the storage for processing user requests for data, as well as on some technical details of the mechanism for selecting the requested data. One of the main points of our approach is the ability to flexibly select data from multiple distributed local storages and seamlessly provide them to users.

The structure of the article is as follows. Section 2 contains a brief description of the AstroDS architecture. In Section 3, we describe the features of the implementation of the mechanism for selecting the requested data. In conclusion, we discuss the results obtained and the plan for the further development of the cloud storage.
2. Design and architecture of the AstroDS

The main principle underlying the development of the AstroDS [3] cloud storage is the principle of maximizing preservation of the historically established methods of user interaction with local storages. Thus, the main requirements were:

- preservation of the structure of data directories;
- an opportunity to mount directories on local computers;
- data transfer over the network should occur at the moment of actual access to them;
- flexible data selection upon user request.

The simplified architecture of the AstroDS is presented in Fig. 1.

![Diagram of the AstroDS architecture](image)

**Figure 1.** The simplified architecture of the AstroDS

The storage consists of the following main components:

- Extractors (E1,E2). These components extract metadata from the input data stream and register it in the metadata catalog service.
- Adapters (A1,A2,A3). These components are gateways for data on the way from local storage to the aggregation service. They are usually implemented using CERNVM FS or simple FTP service.
- Metadata catalog (MDC). This service collects all metadata and is responsible for finding the necessary data upon user request.
- The aggregation service (aggregator). The service provides a user interface and generates data set requested by the user.

A detailed description of the architecture of AstroDS cloud storage can be found in the articles [3, 9].

The AstroDS implements a two-level data selection facilities:

- search at the file level, for example, by session number;
- search at the level of individual events in files, for example, by the energy of the event.

This solution is flexible enough to fulfill almost any user request.
3. Selecting data using the metadata catalog

The Metadata Catalog (MDC) is a service used to determine the physical location of the data with two main functions:

- to register collected metadata;
- to process the user requests for data.

During the processing of a user request MDC examines the collected metadata to determine which local storages and which files contain the data requested by the user.

3.1. Metadata catalog API

The MDC architecture is based on the integration of several standard solutions (see Fig. 2). To store metadata we chose Timescale DB – a special database for storing time-series data. We used the Graphene-Python library [10] to easily create GraphQL APIs in Python to interact with the aggregation service. The SQL Alchemy object-relational mapper (ORM) is used to access the TimeScale DB.

![Figure 2. The architecture design of the MDC](image)

MDC provides an API for metadata insertion and for searching of data location using the filter list shown in Table 1. We do not provide an API for updating metadata in storage and deleting metadata from database because the main idea of AstroDS is that metadata is extracted only once by a special program called an extractor.

The parameters used to select data can be strings, numbers, ranges, or a uniform list of the types mentioned above. For example a user can request the events in which the primary particle energy exceeds 100TeV.

The query structure shown in Listing 1 consists of two main parts - data fields and query parameters. The data fields correspond to the DB schema and include such information as the run date, cluster, weather, facility, etc. All of these data fields are used by the aggregation service to generate a query that is sent to the MDC service. In response, MDC sends URLs to the files that contain the necessary information. Further, the aggregator downloads these files and, if necessary, selects from them those events that meet the selection criteria.
| Parameter name | Description                      | Type        |
|----------------|----------------------------------|-------------|
| fid            | Unique file identifier           | Number      |
| startTime      | Event start time                 | Date        |
| endTime        | Event end time                   | Date        |
| first          | Count of data for pagination    | Auxiliary   |
| offset         | Start position for pagination    | Auxiliary   |
| weatherId      | Weather at the time of observation | Number      |
| trackingSourceId| Observed object                  | Auxiliary   |
| facilityId     | The facility that captured the event | Number      |
| energy         | The energy of primary particle   | Float       |
| partId         | The primary particle identifier  | List of strings |

| Table 1. | Examples of available parameters of user requests. |

Listing 1. The query structure

```javascript
query {
  files ([query parameters]){
    [data fields]
  }
}
```

As mentioned above, the GraphQL library was used for interaction between the aggregator and the MDC. Listing 2 shows an example of MDC response in GraphQL format.

Listing 2. Example of GraphQL response

```javascript
{
  "data": {
    "files": [
      {
        "fid": "98771",
        "fullPath": "/k2/DATA/IACT/2017−18/dec17/201217/BSM08/20127008.029",
        "runDate": "2017−12−20 00:00:00",
        "facilityId": 1
      }
    ]
  }
}
```

3.2. Filters for data selection

The MDC can process data from different facilities which have a lot of parameters for filtering data. The aggregation service gets a list of filters with the formal specifications for each parameter. Parameters can have one of five data types: "date", "int", "float", "string", "list", and "range". The first four data types are basic. The "list" data type could be a query string or an array of base types. The "range" data type is a pair of numbers that define a tolerance interval. One can also set a half-interval. The query string is required when the filtering value is contained in the database table. In this case, the aggregation service makes a request by this query string to the metadata catalog to get a list of parameters.
Specifications are stored in JSON format for each facility. In Listing 3 an example of the specification is shown. The start time and end time are of type "date" and the comparison sign is equal. The third parameter "weather" is of type "list" and contains a query string.

**Listing 3. Filters specification**

```json
{
  "filters": [
    {
      "name": "startTime", "type": "datetime", "conditions": ["="]
    },
    {
      "name": "endTime", "type": "datetime", "conditions": ["="]
    },
    {
      "name": "weather", "type": "list", "options": {
        "table": "weather",
        "request": {"query":{"weatherId":wScale}}
      },
      "fields": {
        "id": {"type": "integer"},
        "wScale": {"type": "string"}
      },
      "conditions": ["="]
    }
  ]
}
```

4. Conclusion
The article describes the functionality of the AstroDS cloud storage for the selection of data upon user request. A prototype of the proposed storage is deployed in Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University. Currently the prototype integrates data from such astrophysical experiments as TAIGA and KASCADE. As a result, users of the respective collaborations already have the opportunity to conduct joint (multi-messenger) data analysis within a wide range of studies in the field of particle astrophysics.

In the future, we plan to expand a list of available filters to increase the flexibility of data selection.

Acknowledgments
Supported by the Russian Science Foundation, grant No.18-41-06003.

References
[1] HESS URL: https://www.mpi-hd.mpg.de/hfm/HESS/.
[2] Rico J for the MAGIC Collaboration 2016 Overview of MAGIC results Nuclear and Particle Physics Proceedings 1273–275, 328–33
[3] Kryukov A et al. 2019 Distributed Data Storage for Modern Astroparticle Physics Experiments CEUR-WS 2406, 78–83.
[4] Budnev N et al. 2017 The TAIGA experiment: From cosmic-ray to gamma-ray astronomy in the Tunka valley. Nuclear Instruments and Methods in Physics Research. Section A 845, 330–33
[5] Hiller R, Budnev N M et al. 2015 Status and First Results of Tunka-rex, an Experiment for the Radio Detection of Air Showers it Phys. Procedia 61, 708-13
[6] Apel W D and et al. 2010 The KASCADE-Grande experiment Nuclear Instruments and Methods in Physics Research Section A 620, 202–16.
[7] KASCADE Cosmic Ray Data Centre (KCDC) URL: https://kcdc.ikp.kit.edu/.
[8] Bychkov I et al. 2018 Russian-German Astroparticle Data Life Cycle Initiative. Data 4 4, 56
[9] Nguyen M-D, Kryukov A and Mikhailov A 2020 The current design and implementation of the AstroDS Data Aggregation Service. CEUR-WS 2679, 106–10
[10] URL: https://graphene-python.org/.