VOTables in Parquet

Progress report — Initial prototyping

Gregory Dubois-Felsmann, Caltech/IPAC
also Andreas Faisst, Jessica Krick, Troy Raen, David Shupe, Brigitta Sipőcz

Motivation

Origins of the work

This project arises from (at least) three paths of work:

- The Rubin Observatory plans to make its major Data Release tables available in a spatially-sharded Parquet format
  - Rubin is very committed to the use of IVOA standards, and by the time the data are released we expect to publish substantial data-model annotations with our data, so we need a Parquet solution
- The LINCC Frameworks project has been working on a cross-project adaptive-depth HEALPix-based scheme for spatially sharding Parquet files together with a software framework (LSDB) for parallelized cross-matching
  - This has been going by a provisional name “hipscat” because it has some structural similarities to HiPS catalogs
- The NASA astrophysics archives have been working on publishing multi-mission catalog data in Parquet in the cloud
  - This is the time-domain connection, because a lot of this work has been on time-series analysis

Motivation

“Parquet is the new CSV” but also “Interoperability is a key value”

- Parquet
  - Efficient I/O, sometimes up to orders of magnitude faster than naive alternatives
    - Column-oriented data storage — read only the columns you need
    - Content-sensitive compression (per-column compression)
  - Powerful open-source tooling for parallel processing
  - BUT very limited standardized metadata on columns (names & datatypes)

- VOTable
  - Lingua franca for exchanging data between services (download & upload)
  - Very extensive standardized metadata - now extended with MIVOT
  - Optimized for interoperability, NOT big-data performance
  - Extensive base of client software

VOTable metadata

What is it?

- Column-specific metadata
  - Extended attributes for columns, beyond hardware datatype:
    - Descriptions
    - Physical units (VOUnit standard, compatible with FITS units)
    - UCDs, providing semantic identification of quantities beyond units (UCD1+ standard)
      - Examples: flux vs. flux error; time vs. time difference; angle vs. coordinate
    - Utypes, associating columns with roles in standardized data models
    - Xtypes, providing labels for complex data types
      - Example: three-element array representing a circle in RA/Dec: [13.4, 4.1, 0.023]
- Data modeling and column grouping
  - MIVOT, traditional <GROUP> usage, etc.
- DataLink service descriptors
  - Document onward queries that can be performed against the data in the table

Proposal

Provide VOTable-style metadata in Parquet

Guidelines

- Preserve ability to use the results with all existing Parquet tooling
- Choose an implementation that eases use in existing IVOA tooling
  - PyVO and other language bindings to the IVOA services
  - UI tools: TOPCAT, Firefly, Aladin, …
  - Server-side infrastructure: enable TAP services to return Parquet on request
    - Support something like RESPONSEFORMAT=Parquet
- Minimize coding effort required & maximize compatibility with the existing, very rich metadata model

Proposal

Implementation concept: use VOTable's XML for the metadata

- Inspired by existing VOTable support for FITS as the data-content format

- Two data formats to be supported
  - Native Parquet: embed VOTable metadata in a Parquet file
    (VOTable XML "headers" stored as Parquet file-level key-value metadata *verbatim*)
  - Use an (XML) VOTable as a "wrapper" to add metadata to an existing Parquet file
    (parallel to what is already supported for FITS)

- Support both single Parquet files and partitioned datasets

After work started on this project, we became aware of the long-standing "FITS-plus" I/O format included in the STIL project, which used very much the same idea to embed VOTable metadata in FITS files.

Proposal

Several modes of use could be supported

Native

1. Incorporation of VOTable metadata into a single Parquet file
2. Incorporation of VOTable metadata into a partitioned Parquet dataset

"Wrapped"

3. Incorporation of a single Parquet file *by reference* into a VOTable as its <DATA> payload, done just as is currently supported for FITS
4. Incorporation of a Parquet partition by reference into a VOTable as its <DATA> payload
5. Incorporation of a single Parquet file *by value* into a VOTable as its <DATA> payload, done just as is currently supported for FITS or BINARY2

Modes 1, 2, and 3 seem the most useful.

Proposal
Native modes

Based on the use of partition-level key-value metadata in Parquet files

- Mode 1:

  The file-level metadata of the Parquet file MUST include these key-value pairs:

  IVOA.VOTable-Parquet.version = 1.0
  IVOA.VOTable-Parquet.type = "Parquet-local-XML"
  IVOA.VOTable-Parquet.encoding = (encoding for the XML content of the following)
  IVOA.VOTable-Parquet.content = (character data in a format valid for an XML file, comprising a single <VOTABLE> element and its hierarchical content)

  The <VOTABLE> must contain exactly one <TABLE> with its <DATA> content specified as follows:

  <DATA><PARQUET type="Parquet-local-XML"/></DATA>

  <FIELD>s map to Parquet columns by ordering just as in existing VOTable

Backwardly compatible with Parquet tools

Existing Parquet tools “just work”

- The VOTable metadata is essentially invisible to existing Parquet tooling and does not interfere with existing community workflows
- But it enables richer, metadata-driven workflows in the future

Prototype Implementation

Status report

- Andreas Faisst has been working on a prototype based on Astropy
  - Troy Raen provided invaluable support with Parquet and with sample data
  - David Shupe and Brigitta Sipőcz provided generous assistance with working with the Astropy organization

- Summary
  - Support for VOTable-XML “wrapper” for existing Parquet (“two-file”, Mode 3) is merged into the development state of Astropy and will appear in Astropy 6.0

  - Support for VOTable metadata embedded in Parquet (“single-file”) has been breadboarded

Progress

Initial support in Astropy

- We have incorporated the “VOTable wrapper” approach in the work toward Astropy 6.0
  - https://github.com/astropy/astropy/pull/15281 has been merged!
- The “embedded header XML” approach has been prototyped but was not ready in time for the Astropy 6.0 cutoff

astropy / docs / whatsnew / 6.0.rst

Preview | Code | Blame | 341 lines (262 loc) · 14.4 KB | Code 55% faster with GitHub Copilot | Raw

VOTable now supports PARQUET serialization

The PARQUET file format allows a more efficient handling of large data amounts. However, one problem of PARQUET is that it only provides a limited number of column metadata keywords. A way to make it consistent with VO standards is to embed it into a VOTable file.

This serialization works similar to the VOTable FITS serialization that already existed. It basically creates two files, on VOTable file and one PARQUET file, which are linked together. The advantage of this method is that any column metadata can be saved along with the PARQUET file, following VO standards.

Reading and writing of the VOTable PARQUET serialization is fully supported by astropy.io.votable and the unified `Table` read/write interface. This serialization can be used by setting the `format` argument to `'votable.parquet'`, while `'votable'` can be used for reading in such a file. The method works for both absolute and relative parquet file paths.

Example for writing:

```
.. doctest-skip::

    >>> import numpy as np
    >>> from astropy.table import Table
    >>>
    >>> # Create some fake data
    >>> number_of_objects = 10
    >>> ids = [f"COSMOS_{ii:03g}" for ii in range(number_of_objects)]
    >>> redshift = np.random.uniform(low=0, high=3, size=number_of_objects)
    >>> mass = np.random.uniform(low=1e8, high=1e10, size=number_of_objects)
    >>> sfr = np.random.uniform(low=1, high=100, size=number_of_objects)
    >>> cosmos = Table([ids, redshift, mass, sfr], names=["id", "z", "mass", "sfr"])
    >>>
    >>> # Create Column metadata
    >>> column_metadata = {
    ...     "id": {"unit": "", "ucd": "meta.id", "utype": "none"},
    ...     "z": {"unit": "", "ucd": "src.redshift", "utype": "none"},
    ...     "mass": {"unit": "solMass", "ucd": "phys.mass", "utype": "none"},
    ...     "sfr": {"unit": "solMass / yr", "ucd": "phys.SFR", "utype": "none"},
    ... }
    >>>
    >>> # Write VOTable with Parquet serialization
    >>> filename = "votable_with_parquet.vot"
    >>> cosmos.write(filename, column_metadata=column_metadata, format="votable.parquet")
```

Road Map

Next steps in bringing this out to the community for feedback

- Mode-3 support is in Astropy pre-6.0
- We will aim to get “native” (Modes 1, 2) support into the next major release

- IPAC will look into working Parquet-reading into the Firefly development road map

- Planning to submit a Note on the work, especially the “native” modes (1, 2) in the near future
  - A draft has been circulated to a few IVOA-community people for feedback already

- Expect to adopt the native mode(s) for use in the Rubin Observatory’s spatially-partitioned catalogs