Implementing, with Python and PostgreSQL, Virtual Observatory services for publishing survey data from the OAJ

Javier Hernández

CEFCA
CEFCA and the OAJ

The Observatorio Astrofísico de Javalambre (OAJ, Teruel, Spain)

CEFCA is an institution of the Government of Aragón for research in Astrophysics and Cosmology, whose activities focus on the technological development and operation of the Observatorio Astrofísico de Javalambre (OAJ, Teruel, Spain) and on its scientific exploitation.
Two main telescopes with large fields of view (FoV) and image quality all over their entire FoVs:

- The 80cm Javalambre Auxiliary Survey Telescope, JAST/T80, with a FoV of 2 deg.
- The 2.5m Javalambre Survey Telescope, JST/T250, a large-etendue telescope with a FoV of 3 deg diameter.

The scientific instruments for the telescopes are:

- For the JAST/T80 a wide-field camera equipped with a 9.2k-by-9.2k high efficiency CCD. The Filter Unit holds two removable filter wheels holding 12 filters.
- JPCam is a wide field 14 CCD-mosaic camera that for the JST/T250. The Filter Unit admit five filter trays with 14 filters each.
The Javalambre-Photometric Local Universe Survey, J-PLUS, is a photometric sky survey of 8500 deg$^2$, using a set of 12 broad, intermediate and narrow band filters. Designed to carry out the photometric calibration of J-PAS.

Javalambre Physics of the Accelerating Universe Astrophysical Survey, J-PAS, is an unprecedented photometric sky survey of 8500 deg$^2$ in 59 colors, using 54 narrow plus 5 broad bands.
Portal OK but *not enough appropriate* for:

- Bulk data processing.
- Automation.
- Interoperability.

Virtual Observatory (VO) comes to scene:

- The IVOA (www.ivoa.net) Virtual Observatory promotes **standards** that allow seamless access to data.
- There is an increasing number of tools supporting them.
- Well suited to bulk access and automation scripts.
Choosing VO Services

- Simple Image Access: image search and download (full and *cutouts*).
- Simple Cone Search: catalogue object search.
- TAP: search images, catalogue objects, derived data (*photo-redshifts*, stellarity, ...). Describe data.
- SAMP: push data from our web portal.
Our own server side implementation

- There were available toolkits to publish data like DaCHS from GAVO, or SVOCat from Spanish VO. ([https://wiki.ivoa.net/twiki/bin/view/IVOA/PublishingInTheVO](https://wiki.ivoa.net/twiki/bin/view/IVOA/PublishingInTheVO))
- But they did not fulfil all our needs: integration with our portal, access control for certain catalogue versions, and support for a few custom extensions (*more later*...).
- IVOA services are mainly based on *web technology*, and in general they are simple.
We wanted an open source SQL database (*ADQL* friendly).

J-PLUS and J-PAS Catalogue final data is expected to occupy several terabytes.

With flux and error measures for about 16 apertures in 59 filters, just for photometry, we have 1888 columns of data in the object table.

**PostgreSQL** has known support of big databases, support for custom functions, and *array types* (which simplify storing all filter flux data in one column).

Spatial index in the database is very important, we opted to use **Healpix**.
The Python programming language was in use in the house for the reduction pipelines, so we evaluated it for creating web applications.

The Python WSGI specification, with a lot of implementations (Apache mod_wsgi, Unicorn, uWSGI, ...), supports web applications.

A lot of frameworks (Pyramid, Flask, Django, ...) simplify development, easily map a Python function to a web URL, and access to web parameters.

Large amount of libraries for astronomy, database access, and Healpix are available.
And some results ...

SAMP with Aladin. TAP web interface. TAP Topcat.
A few lessons learned

- TAP more difficult than we thought. The idea to pass directly queries to the database was not feasible, needed a SQL dialect translation. But good for security, limiting result size or add our extensions (enumerations to assign names to filter positions ‘jplus::rSDSS’).

- Also in TAP we found that implementing the geometric functions is very complex due to the rich functionality defined, so at the moment only partial support for that functions exists.

- IVOA centres on public data so access control is not standardised. We finally achieved to support authentication for some tools like Topcat using Basic HTTP authentication.
Performance is always something that at some point you have to improve.

We initially implemented in Python the needed database functions for ADQL but later we moved to a C implementation because it is ten times faster.

TAP queries can take some minutes to execute, so executing concurrently them is mandatory. Python threads have some blocking issues, fortunately Python multiprocessing package has a similar API.
Thank you!