pyVO now does TAP!

Stefan Becker[1,2], Markus Demleitner[2]
[1] SFB 881 "The Milky Way System" [2] Universität Heidelberg, Astronomisches Rechen-Institut, Mönchhofstraße 12-14, Germany
sbecker@ari.uni-heidelberg.de

Abstract

PyVO is an astropy-affiliated package providing an API for the access and retrieval of astronomical datasets from the Virtual Observatory (VO) using various VO Data Access Layer Protocols. We have recently added support for the Table Access Protocol (TAP) in pyVO. With this, pyVO now supports synchronous and asynchronous queries, including the upload of local tables. PyVO's TAP support also allows inspection of the service metadata. Thanks to astropy integration, it is straightforward to work with the results obtained and re-use them either in further VO queries or in custom python code.

Basic TAP API

Module Import, Service Creation: Most TAP funtionality is provided through a `TAPService` object constructed with the base access URL of the TAP service:

```
from pyvo.dal import tap

service = tap.TAPService(
  "http://dc.g-vo.org/tap")
```

Synchronous Query: A basic synchronous query is run by passing the query string to a service's `run_sync` method. The query result is available in the return value's `votable.to_table()` attribute (this is for compatibility with the rest of pyVO).

```
result = service.run_sync(
  """SELECT ROUND(rv/5) AS bin, al, si, fe, fe_n
  FROM rave.main
  WHERE rv BETWEEN 40 AND 70 AND al IS NOT NULL"""
).votable.to_table()
```

Job Customisation: Most TAP services enforce relatively small match limits when not giving TAP's MAXREC argument. You can explicitly pass it:

```
result = service.run_sync("SELECT * FROM ppmxl.main",
  maxrec=1000000
).votable.to_table()
```

(language works analogously).

Uploads: You can use local tables in your remote queries:

```
from astropy.table import Table
[...]
local_table = Table([ras, decs, pmras, pmdecs],
      names=("ra", "dec", "pmra", "pmdec"))
response = service.run_sync(
  """SELECT r.* FROM remote as r
    JOIN TAP_UPLOAD.t1 as l
    ON 1=CONTAINS(
      ivo_apply_pm(l.ra, l.dec, l.pmra, l.pmdec, -15),
      CIRCLE('', r.raj2000, r.dej2000, 1/3600.))""",
  uploads={"t1": ('inline', local_table)}
).votable.to_table()
```

Async TAP API

For long-running jobs, TAP lets clients execute jobs asyncronously, using the UWS job pattern.

Simple async querying: For simple cases, pyVO has a simple wrapper around the UWS interaction that is signature-compatible with the synchronous method:

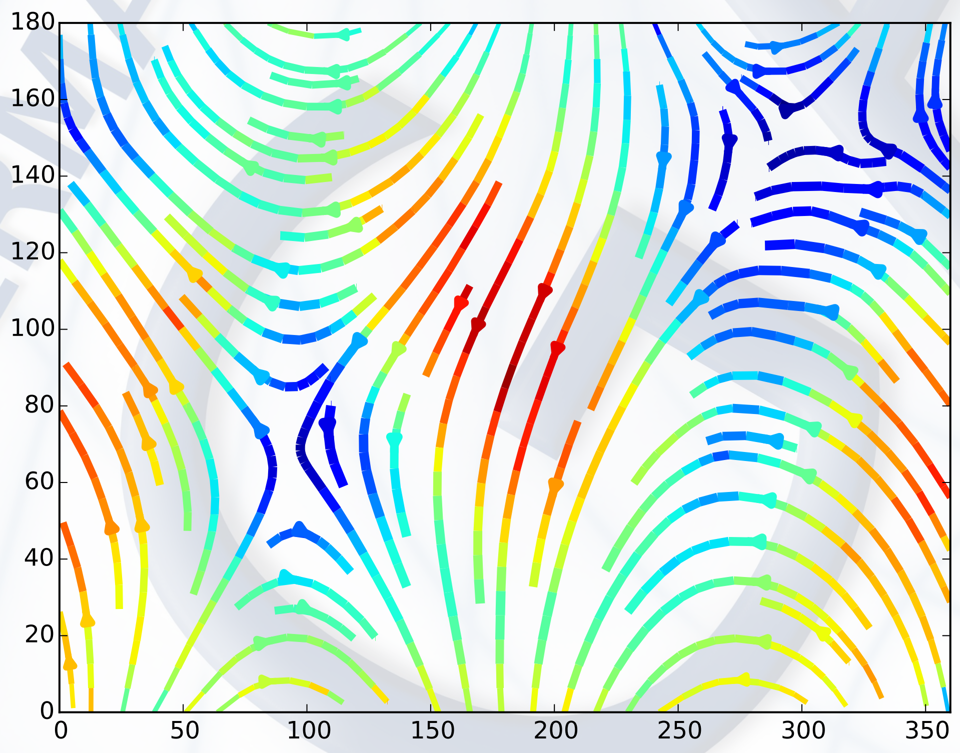
```
result = service.run_async(
  """SELECT ROUND(rv/5) AS bin, al, si, fe, fe_n
  FROM rave.main
  WHERE rv BETWEEN 40 AND 70 AND al IS NOT NULL"""
).votable.to_table()
```

Adanced Async Operation: For more advanced scenarios, you can submit a job and receive a non-started **AsyncTAPJob** object:

```
job = service.submit_job("SELECT * FROM gaia.dr1")
job.execution_duration = 3600*48
job.destruction = datetime.datetime.utcnow(
  )+datetime.timedelta(days=4)
job.run()
print("Come back in two days and resume %s"%job.url))
```

Resuming Async Jobs: In the above scenario, the program would exit and let the remote job run. To pick it up again, one would obtain the job URL just printed and run something like:

```
job = tap.AsyncTAPJob(job_url)
job.wait()
job.raise_if_error()
result = job.fetch().votable.to_table()
```



The figure to the left was produced with the following pyVO program:

```
import matplotlib.pyplot as plt
import numpy as np
import pyvo

BIN_SIZE = 10

def clip(arr, limit):
    arr[arr>limit] = limit
    arr[arr<-limit] = limit

svc = pyvo.dal.TAPService("http://dc.g-vo.org/tap")
res = svc.run_sync(
    """SELECT
        round(raj2000/{}) AS xind,
        round((dej2000+90)/{}) AS yind,
        avg(pmra) as pmra, avg(pmdec) as pmde,
        count(*) as ct
        FROM tgas.main
        GROUP by xind, yind""".format(BIN_SIZE, BIN_SIZE),
    maxrec=1000000).votable.to_table()

clip(res["pmde"], 50)
clip(res["pmra"], 50)

w, h = max(res["xind"]), max(res["yind"])
u, v, weight = [np.zeros((h+1, w+1)) for _ in "123"]
for x, y, pmra, pmde, ct in res:
    u[y, x], v[y, x] = pmde, pmra
    weight[y, x] = np.log(ct)
weight = 5*weight/np.max(weight)

plt.streamplot(np.arange(w+1)*BIN_SIZE, np.arange(h+1)*BIN_SIZE,
    u, v, color=np.sqrt(u*u+v*v), linewidth=weight)
plt.xlim(0, 360)
plt.savefig("explot.eps", format="eps")
```

Installation

TAP support entered pyVO in version 0.3, which will be available on pyPI soon. You can then obtain it by simply using `pip install pyvo`.

Meanwhile, please try our code by cloning `https://github.com/pyvirtobs/pyvo.git`. We also appreciate bug reports or feature requests on github.

Acknowledgement: This work was supported by Sonderforschungsbereich SFB 881 "The Milky Way System" of the German Research Foundation (DFG).