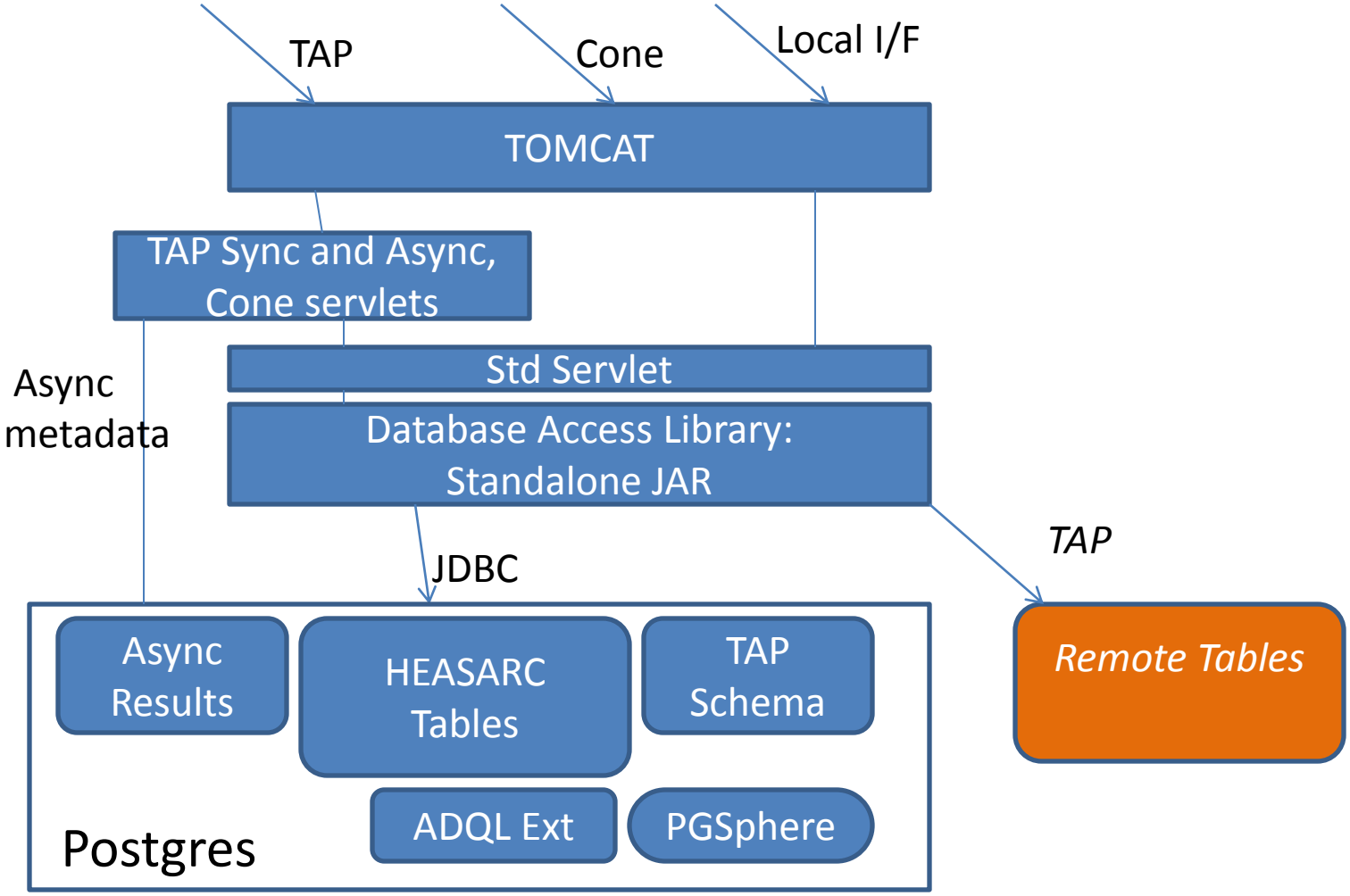
Using TAP at the HEASARC

Tom McGlynn

HEASARC Web Access



HEASARC Overview

- 600 Tables
- 40 million total rows
- 20,000 total columns
- Index of ~20 TB archive

TAP Implementation

- Capabilities
  - ADQL
    - Minimal region capabilities
    - Full support for other geometry capabilities
  - PQL (no multicone or region support)
  - VOSI (excluding getCapabilities)
  - TAP Schema
  - Sync + Async
  - No version negotiation

ADQL

- Add direct support for ADQL to Postgres
  - PgSphere
  - Limited region support
    - No parser to convert regions to Circle/Box/Polygon objects
  - Works acceptably but efficient queries of large tables require user to add indexible spatial constraints

PgSphere usage

- Two implementations of ADQL geometry
  - Compound geometry object on top of PgSphere objects: geometry constructs return new type.
  - Direct use of existing PgSphere types
- Easy and fully functional.
- Issues
  - Output format (geometry objects in select clause)
  - Making effective use of indexes

```
-- Create a point using the ADQL syntax
create function point(text, float8, float8) returns
    spoint as $$
    select case when lower($1) = 'icrs geocenter' or
                    lower($1) = 'j2000' or
                    lower($1) = 'icrs' or lower($1) = 'fk5'
                 then
                    spoint(radians($2),radians($3))
                else
                    spoint(radians($2),radians($3)) +
                      stdTrans($1)
end;
$$ language SQL;
```

PQL

- Straightforward translation of PQL arguments to local arguments
- Simple parser for PQL Where clause
- No multicone yet but should be easy to implement.

Async/UWS implementation

- Create TAP_Async schema in Database
- Use clock to define get ID (OK to start)
- Copy results to table in new schema using job ID in table name.
  - Currently store table, but maybe better to store blob of text output. I.e., go from one table per job to one row.
- Async metadata in Postgres tables.
- No direct use of file system for Async.

TAP Schema

- Just created Postgres Schema TAP_Schema
- Create tables exactly as defined in standard
- VOSI /tables uses same tables.
- Generated using standard query on underlying metadata tables (very different format).

Example URLs

- HEASARC Query
  http://heasarcdev.gsfc.nasa.gov/xamin/query?
  position=3c273&table=rosmaster&constraint=exposure>10000
- Cone Search Query
  http://.../xamin/vo/cone?table=rosmaster&RA=187.25&DEC=2.05&SR=1
- TAP Sync PQL Query
  http:…/xamin/vo/tap/sync?
  language=PQL&REQUEST=doQuery&table=rosmaster&
  where=exposure,10000/&POS=187.25,2.05
- TAP Sync ADQL Query
  http://xamin/tap/sync?
  language=ADQL&REQUEST=doQuery&
  query=select+*+from+rosmaster+where+exposure>10000+and+
  Contains(point('j2000', ra,dec),circle('j2000', 187.25,2.05))=1

Evil script to test Async

```
#!/usr1/local/bin/perl
 wget("create", '?request=doquery&lang=pql&table=messier');
 open(IN, "create.out");
 my $data = "";
while (<IN>) {
   $data .= $_; }
close(IN);
$data =~ /<jobId>(.*)<\/jobId>/m;
my $id = $1;
$id =~ s/(^\s*|\s*$)//;
print "Job id is: $id\n";
wget('start', "/$id/phase?phase=run");
sleep 2;
wget('check', "/$id");
wget('results', "/$id/results/result");

sub wget {
    my ($file, $suffix) = @_;
    my $cmd = "wget -O $file.out “.
     “'http://heasarcdev.gsfc.nasa.gov/xamin/vo/tap/async$suffix'";
   print "Sending cmd:\n $cmd\n";
   `$cmd`;
}
```

Script output

Sending cmd:
 wget -O create.out 'http://heasarcdev.gsfc.nasa.gov/xamin/vo/tap/async?request=doquery&lang=pql&table=messier'
Job id is: 1257892929258
Sending cmd:
 wget -O start.out 'http://heasarcdev.gsfc.nasa.gov/xamin/vo/tap/async/1257892929258/phase?phase=run'
Sending cmd:
 wget -O check.out 'http://heasarcdev.gsfc.nasa.gov/xamin/vo/tap/async/1257892929258'
Sending cmd:
 wget -O results.out 'http://heasarcdev.gsfc.nasa.gov/xamin/vo/tap/async/1257892929258/results/result'

Create response

<job>
<jobId>1257892929258</jobId>
<ownerId/>
<phase>PENDING</phase>
<startTime>Tue Nov 10 17:42:09 EST 2009</startTime>
<endTime>Wed Nov 11 17:42:09 EST 2009</endTime>
<executionDuration>600000.0</executionDuration>
<parameters>
<parameter id="table">messier</parameter>
<parameter id="lang">pql</parameter>
</parameters>
<ErrorSummary/>
<results>
<result id="resultsTable"
xlink:href="xamin/vo/tap/async/1257892929258/results/result"/>
</results>
</job>

Start response

<job>
<jobId>1257892929258</jobId>
<ownerId/>
<phase>EXECUTING</phase>
<startTime>Tue Nov 10 17:42:09 EST 2009</startTime>
<endTime>Wed Nov 11 17:42:09 EST 2009</endTime>
<executionDuration>600000.0</executionDuration>
<parameters>
<parameter id="table">messier</parameter>
<parameter id="lang">pql</parameter>
</parameters>
<ErrorSummary/>
<results>
<result id="resultsTable“
xlink:href="xamin/vo/tap/async/1257892929258/results/result"/>
</results>
</job>

Check response

<job>
<jobId>1257892929258</jobId>
<ownerId/>
<phase>COMPLETED</phase>
<startTime>Tue Nov 10 17:42:09 EST 2009</startTime>
<endTime>Wed Nov 11 17:42:09 EST 2009</endTime>
<executionDuration>600000.0</executionDuration>
<parameters>
<parameter id="table">messier</parameter>
<parameter id="lang">pql</parameter>
</parameters>
<ErrorSummary/>
<results>
<result id="resultsTable"
xlink:href="xamin/vo/tap/async/1257892929258/results/result"/>
</results>
</job>

<VOTABLE
xsi:noNamespaceSchemaLocation="http://www.ivoa.net/VOTable/VOTable/v1.1"
version="1.1">
<RESOURCE>
<TABLE>
<DESCRIPTION>
Query result for:
select name, alt_name, ra, dec, constell, dimension, vmag, vmag_uncert, class from
tap_async.t1257892929258
</DESCRIPTION>
<FIELD name="name" arraysize="*" datatype="char"/>
<FIELD name="alt_name" arraysize="*" datatype="char"/>
<FIELD name="ra" datatype="double"/>
<FIELD name="dec" datatype="double"/>
<FIELD name="constell" arraysize="*" datatype="char"/>
<FIELD name="dimension" arraysize="*" datatype="char"/>
<FIELD name="vmag" datatype="double"/>
<FIELD name="vmag_uncert" arraysize="*" datatype="char"/>
<FIELD name="class" datatype="long"/>
<DATA>
<TABLEDATA>
<TR>
<TD>M 7</TD>
…