Monitoring VO Service Performance in NAVO

Tom Donaldson

IVOA Interoperability Meeting

November, 2020

What is NAVO?

“The NASA Astronomical Virtual Observatories (NAVO) program coordinates the efforts of NASA astronomy archives in providing comprehensive and consistent access to NASA's astronomical data through standardized interfaces. “

- Mikulski Archive at Space Telescope (MAST)
- High Energy Astrophysics Science Archive Research Center (HEASARC),
- NASA/IPAC Infrared Science Archive (IRSA)
- NASA Extragalactic Database (NED)

https://heasarc.gsfc.nasa.gov/vo/summary/navo_intro.html

Why monitor service performance?

"Is that TAP query taking longer than usual?"

"The catalog isn't that big. Why does my cone search take forever?"

Users' perception of performance is anecdotal and vague.

- Repeatedly sampling the performance of services gives a better idea of what is typical, allowing
  - Detection of unexpected performance regressions
  - Comparison with new technologies (cloud DBs, no-sql, etc.)
  - Identification of services that
    - Perform significantly worse than peer services
    - Or simply don't perform well enough

- As our data holdings increase in size, performance gets more and more important

Getting Started

The team:

- Bruce Berriman, John Good (Caltech/IPAC)
- Tom Donaldson, Bernie Shiao (STScI)

Started by studying the comparative performance of databases with these variables:

- Indexing depth (cell size)
- Hierarchical Triangular Mesh (HTM) vs. HEALPix (HPX)
- PostgreSQL vs. SQL Servers
- Linux vs. Solaris vs. Windows

Conclusions

- Total query time was dominated by I/O.
- Indexing depth—and not choice of index—has the greatest impact on performance

So we widened the scope beyond the DBs to the actual VO Services…

What queries are we measuring?

Services: Cone Search, Simple Image Access and TAP (sync and async)

- NAVO archives
  - Select collections with representative coverage and densities
- More limited queries for baseline comparisons at
  - Chandra (source catalog)
  - CDS and GAVO (2MASS)

Cones: Randomized list of positions and radii (<0.25 deg)

For each service, query each cone

- TAP queries are done on the cone region
- Queries done in series (per institution), not parallel
  - Do not want to impact operations
  - Not load testing, just measuring individual queries
    - Other server load obviously must be considered, but may even out over time
    - Future work may do load testing on isolated NAVO services

What data are we collecting?

Query metadata

- Service provider and access_url
- Query parameters (position, radius, ADQL)

Timing numbers

- Time to first response
- Time to stream results
- Time to make Async TAP calls
  - Submit
  - Run
  - Wait

Result metadata

- HTTP response code (and error message, if any)
- Size in bytes
- Number of rows and columns

How did we do the queries?

Developed a Python package called *servicemon*

- https://github.com/NASA-NAVO/servicemon
  - Open development: Feedback, issues, pull requests welcome
- https://servicemon.readthedocs.io/en/latest/

great_archive_tap_service.py

```
[
  {'base_name': 'great_archive_tap',
  'service_type': 'tap',
  'access_url': 'http://services.great_archive.net/tap_service',
  'adql':'''
  SELECT *
  FROM ivoa.obscore
  WHERE
    1=CONTAINS(POINT('ICRS', s_ra, s_dec),
               CIRCLE('ICRS', {}, {}, {} ))
  '''
  }
]
```

Features

- Callable by command line or Python API
- Generate random cones (positions and radii)
- Query endpoints defined Python dictionaries
- Query multiple archives in parallel
  - Determined by directory structure of query definition files
- Plug-in infrastructure allows custom output handling
  - Built-in CSV writer provides default handling
  - Actively being extended to support custom queries and randomized non-cone input

Why not use Locust?

LOCUST

"Locust is an easy to use, scriptable and scalable performance testing tool."

- https://locust.io
- Open source with many users and a Slack space for support
- Core idea is to simulate many simultaneous users
  - "Swarm" the service (like locusts)
- Customizable

Initial Analysis

- Features geared towards load testing, which was not our goal
  - Working around some behaviors seemed a little kludgy
- Wanted to establish our query mechanism and data collection without constraints
  - Experiment with different levels of timing granularity
  - Gather VO-specific result metadata

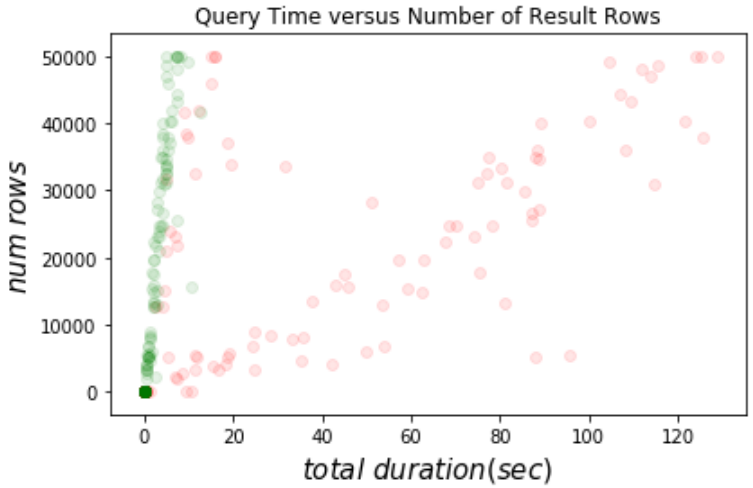
Might revisit, esp. for load testing use cases

- Likely could use customization features to include servicemon code/functionality

Early Findings Useful

Able to reliably gather baseline data

- Most services responded in reasonable time.
- Results were consistent over a few samples of thousands of queries



But in some cases we found examples of:

- Surprisingly slow services
- Surprising variability in response times
- Service bugs

- Oddities like asynchronous TAP taking much longer than synchronous
- PyVO quirks
  - More http requests than needed for async TAP
  - Gathering TAP timing information requires surgery on PyVO functions
    - Might contribute generalized modifications back to PyVO to support instrumentation like this, but PyVO has multiple layers of encapsulation that discourages that idea

Setting Up Operational Monitoring (in progress)

Servicemon in AWS

- Run from several locations in the cloud
  - Limit location bias
  - Identify issues for more remote users
  - Initial cost estimates are quite low (< 50 USD per server per year)

Handling Results

- Use servcemon plugin mechanism to POST results to centralized NAVO server
  - Starting by writing JSON to MongoDB instance at IPAC
- Result DB will have query API
  - Developing basic web UI/dashboard for quick looks
- Will eventually merge with other NAVO monitoring data
  - https://heasarc.gsfc.nasa.gov/vo/monitor/bin/perl/vodb.pl
  - https://heasarc.gsfc.nasa.gov/vo/monitor/stats.html

Potential Interest Outside NAVO?

Once this is well-established in an operational setting, might it be useful/desired to gather statistics for non-NAVO institutions?

Parameters to consider:

- Which services to monitor
- Impact on services
  - Wouldn't require a huge number of queries, but need
    - no impact on service performance
    - correctable impact on service logs (e.g., filtering by User-Agent)
- Number and frequency of queries
  - Delay between individual queries desired?
- Query parameters (size of radius, etc.)
- Political sensitivities
- Others?