52 North SOS Server improvements

Geoff Williams

Australian Government

Bureau of Meteorology

Disclaimer

- I am not an Employee of the Bureau of Meteorology or CSIRO and do not represent them
- ...But am grateful to them sponsoring the body of work I've done so far!

Background

- We wanted to use SOS to deliver historical data from large datasets – millions of observations from thousands of sites
- ...But in its present state the SOS 3.5 trunk code just couldn't handle this amount of data

Image courtesy of Flickr user:
http://www.flickr.com/photos/60258236@N02/

Problems faced

- Attempting to bulk-load data using SOS-T
  - Got slower as more data was loaded
- Startup speed depends on amount of data
  - With enough data loaded, the server NEVER starts!
- Memory requirement at idle is dependent on amount of data – gigabytes + see above
- Whole response is build as a java data structure (XMLBEANS) before being sent to client – also gigabytes
- Configuration needs to be “compiled” into the .war file – we end up custom builds for each environment

The bottom line

- It is a requirement for our team to deliver data using SOS
- Our IT department looked at the current peformance requirements* and said: **“no way!”**

** Requirements currently depend on the amount of data your serving – small datasets are fine, large ones less so*

What to do now!?

- Have computer; will code!
- Got in touch with the 52 north team and our partners at CSIRO to discuss how to fix things
  - Very approachable and open to new ideas

Key Approaches

- Only keep data in memory when it is **needed immediately**
- Do not **block** operations unnecessarily
- Keep code as **simple** as possible – use Spring Framework to replace custom introspection and error handling
- Build and send data in **small chunks** instead of complete documents
- ...And above all: **Do as little as possible :-D**

Streaming

The basic approach is:

```
...
try {
    while (sosResponse.hasMoreDataChunks()) {
        bytes = sosResponse.getNextDataChunk();
        out.write(bytes);
    }
    sosResponse.close();
} catch (IOException ioe) {
    // client hangup
    sosResponse.close();
}
...
```

Additional output formats

- Rewritten OutputFormat support makes adding new output formats easy
- New output formats are registered using springs @Component annotation and implementing IoutputFormat (sos-coding-csv module is a good template).
- No need to alter the core code/XML files/properties flies!
- ...Anyone for `image/png`?

Major changes

- Table scans on startup execute in a background thread and don't block startup
- Use spring beans instead of config file + introspection to load listeners
- Remove static classes + singletons and replace with spring managed singletons
- Remove big lists from memory (CapabilitiesCache) -- lookup values directly
- Pregenerate Capabilities xml fragments and database aggregate statistics and store in a H2 database using Hibernate.
- Stream output for GetObservations and GetCapabilites using blocking IO
- Support streaming for CSV and WaterML2 output formats
- Externalise config file - Config file can now be kept outside .war file - server knows about its location from JVM argument -DN52_SOS_CONFIG_DIR or corresponding JNDI variable in web.xml (if set).
- Rewrote exchangeable encodings code to allow self-registering components to be loaded (without introspection)
- Coded support for caching the capabilites output via string concatenation in a stringbuilder rather then using XML beans (memory)
- Coded support in PG*DAOs to perform some XML generation directly in postgres for value lists
- Use spring JDBC to reduce code for handling sql exceptions in pgconfigdao
- Optimise the SQL used for building the cache to use a recursive postgres CTE to resolve nested procedure hierarchies.

Minor changes

- Security - when serving sensorml files, only serve them from a given directory
- Optimise the iso_timestamp PLPGSQL function to not use XML functions (slow) and to always return times at UTC+0
- Service and provider information in XML fragments is interpolated from config file at runtime
- Logback.xml file can be kept in config directory and will be loaded if it exists
- Security - added switch to sos.config to disable SOS-T for production environments
- Set properties directly in sos.config and dssos.conf - no more build.properties
- Fair queuing/DOS - extra config file variable to limit concurrent getcapabilities requests.
- Test client pages are now JSP processed and lookup the service url at runtime
- Ease of testing/proxying - added a parameter to sos.config to allow invalid URL parameters to not cause an exception
- Created servlet at /statistics_cache_status to give visibility to the state of the cache update thread
- Merged SISS teams support for WML2 DR -- ask for response format "http://www.opengis.net/waterml-dr/2.0" in sos 2.0.0 to trigger
- Detect when client aborts connection and stop processing requests
- Removed gzip encoding in favour of the container/http server performing this task for us (mod_deflate) -- send an accept-encoding = "gzip" request to activate
- Catch java errors to prevent returning raw stack traces to users (now get a service exception or raw "error" if response is streaming
- ... Probably other stuff I've forgotten

Remaining work

- XML Validation
  - GetCapabilities 2.0.0 -- not currently validating to schema
  - WML2 TVP -- not currently validating to schema
  - Check GetObservation output is valid after streaming large response (check that limit + offset haven't inadvertently corrupted the datastream)
- Under very heavy load GetCapabilities requests seem to permanently 'stick'. I believe something goes wrong with the connection to H2 and it blocks forever, breaking GetCapabilities but still allowing data requests to succeed
- SOAP compatibility is currently broken (write out to file; flush file to client)
- TESTING
  - Run test suite (- I believe one exists now?)
  - I've tested the services I actively use, the way I use them. Corner cases (eg yours) might break things
- Tidying up
  - Licensing/copyright headers
  - Comments

Ultimate Goal

- **Merge** this code back with the 52 North source repository
  - It started off as a patch but now its huge
- Run SOS server on modest hardware
- Public facing **deployment** of 52 North SOS server at the Australian Bureau of Meteorology

Questions/Want to help?

Geoff Williams

- geoff@geoffwilliams.me.uk

Thanks to the Austrlian Bureau of Meteorology for sponsoring this work and to 52 North and CSIRO for helping:

Australian Government

Bureau of Meteorology