



SIAP 1.0 Implementation

Overview

We have implemented several SIAP 1.0 compliant services to deliver high quality, interesting public imaging data from CADC data collections. Our SIAP services provide a large number of images (~500,000) across a range of wavelengths (sub-mm, optical, and UV). We support extraction of extensions from multi-extension FITS (MEF) files, image cutouts, and WCS correction. Image query performance is very good (typically ~1 second to query a few square degrees and find several thousand images that intersect the region of interest).

This report describes the implementation of CADC SIAP services from a software and technology point of view.

Major Components

WcsInfo

We are in the process of computing and storing WCS parameters in archive databases. In most cases, we analyse the images and compute corrected WCS params whenever possible. This work is archive-specific; each archive maintains one or more database tables to store WCS parameters for its datasets.

Archive Views

The standard practice is to create one or more views within each archive database; the views are responsible for collecting all SIAP-relevant parameters together in a standard form. The relevant parameters include the spatial description (WCS params), spectral description (filter names and representative wavelengths), temporal description (observation date and exposure time), and data access params (archive name, file identifiers, and optional data transformation operations). For our purposes, it also includes some conditions like only showing public data and only showing data where the WCS information is of acceptable quality. The view(s) insulates the loading software from the details of the archive organisation.

Spatial Database Loader

We have implemented one program to load image descriptions from archive views into a database optimised for the SIAP image query (region-of-interest vs. image bounds intersection). The spatial database contains all required and recommended

SIAP 1.0 values, although we allow spectral and temporal values to be null.

In DB2, we chose the simple but effective approach of placing the corners of the image onto a unit sphere (using WCS to convert to longitude and latitude) and then storing and indexing the bounding cube using Multi-Dimensional Clustering (MDC); this is essentially a 6-dimensional index. The MDC indexing mechanism in DB2 is a highly optimised scanning technique which has the effect of making multi-dimensional queries bandwidth-limited rather than seek-time limited (the more typical case in standard relational database systems).

SIAP Image Query

We implemented an image query servlet in Java to perform the SIAP query against the spatial database and write the result as a VOTable. We only support two levels of verbosity for the image query. As described in the SIAP 1.0 specification, our default verbosity level is 1; at this level we return all the required fields. The other level we support is level 3; At this level we return all the required and optional fields with UCDS as described in the specification. We do not return anything where we made up or chose our own UCDS.

To query the spatial database, the region-of-interest (ROI) is transformed into a cube (in the same way that the spatial bounds of the image was during loading) and some straightforward SQL is generated for the intersection of two cubes. Depending on the size of the ROI, about 10 to 50% of the entries returned by the query don't actually intersect. When the ROI is much larger than the typical image size, we see about 10% false hits; when the ROI is comparable to the typical image size, we see about 30-50% false hits, but with considerable variation. To improve the value of the query output, we perform clipping using the WCS parameters to find the subsection of the image that actually intersects the ROI. With the clipping, we only return images with some pixels in the ROI. Furthermore, we now know exactly which pixels are within the ROI and can use that to do an image cutout during data retrieval (see below). For some data collections, the images are not large and we do not bother with cutouts (SCUBA images from JCMT, for example). Also, if the ROI is large enough that most of the image would be returned, we do not bother with the cutout. We always use clipping to make the spatial query exact, but cutouts are not always performed.

We are currently using the VO-India streaming VOTableWriter to help in writing the VOTable output from the image query. The big advantage is that we don't have to create and hold the entire VOTable structure in memory during conversion to XML. The downside is that this package allows one to create non-well-formed XML documents and certainly doesn't help in creating documents that are valid with respect to the VOTable schema and the requirements of the SIAP specification. Using this simple package puts an unacceptably large burden on testing. More importantly, the value of XML is that XML parsers must enforce well-formedness and it is

straightforward to test for validity when receiving an XML document. It seems clear that we should have used a package that guaranteed the production of valid VOTable documents.

Data Access Proxy

The last major component is the data access proxy (servlet) that can decode the SIAP AccessRef value and return the data (a simple FITS file in this case). Users can see the URL to the proxy and can put in any values for the data access params they like. Rather than try to obfuscate or limit the use of this retrieval mechanism, we implemented a proxy that can deliver any file in the archive; because CADC data collections include both public and proprietary data, we also had to implement an authorization mechanism to enforce access rights.

The proxy also performs some data transformations (extension extraction for MEF files, image cutouts where appropriate, and correction of the WCS params in the FITS header). We currently deploy proxies with 3 different configurations:

The `/ivoa/getData` proxy allows anonymous access and always performs WCS correction; since it gets the WCS parameters from the spatial database, it can only deliver images whose descriptions have been loaded into the spatial database (see above).

The `/anonProxy/getData` proxy allows anonymous access and lets the user choose original or corrected WCS parameters. This proxy can deliver any public file in the archive.

The `/authProxy/getData` proxy requires the user to authenticate themselves (currently using basic http authentication); this proxy uses the supplied authentication to determine if the user has access rights for the requested file, so proprietary data can be retrieved. In all cases, we log the username and reason (an access authorization rule) a download was permitted. (For the anonymous proxies, we log data retrievals as 'voUser' and 'anonUser' respectively, so we can tell roughly what brought the user to CADC but not who they are. For the authenticating proxy, we log the real username.)

Since we have a distributed storage system and had to deal with MEF files, we had two choices: copy large files to the proxy and extract extensions there or support the ability to do simple file manipulation on the storage server. We adopted the latter approach by allowing for plugins (arbitrary file operations) to be implemented and added to the storage system and then just request certain operations to be performed during retrieval. This has several advantages. Header extraction, extension extraction, and image cutouts happen when the file is still on random access storage (disk) which allows a well-implemented plugin to reduce latency. Reducing latency is actually more important than the savings in network traffic since we have Gbit between the storage server and proxy. Second, the data can be streamed from the storage system to

the user, passing through various transformations along the way but never staged (stored on disk); this removes most of the resource contention issues and makes for a more robust system, especially under heavy load.

Notes

External Technologies

CADC SIAP services and tools were implemented using Java , Sybase (archive databases) and DB2 (clustered spatial database). The SIAP query servlet and data retrieval proxies are deployed using Apache Tomcat on a pair of Linux web servers. This allows for load balancing query and data retrieval activities; system failures and maintenance will generally not result in the unavailability of the service. The WCS correction work within the archive is implemented in an archive-specific fashion.

Issues

The main issue we ran into with SIAP is that large mosaic camera data from CFHT is stored as multi-extension FITS (MEF) format. We decided to split MEF files into one logical SIAP image per extension and thus lose information about the large field-of-view of the mosaic cameras. This is needed at a lower level anyway because the WCS solutions are computed on a per-extension basis anyway and we do not want to resample the mosaic into a single large image.