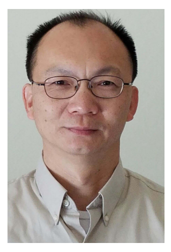
Large Synoptic Survey Telescope

www.lsst.org



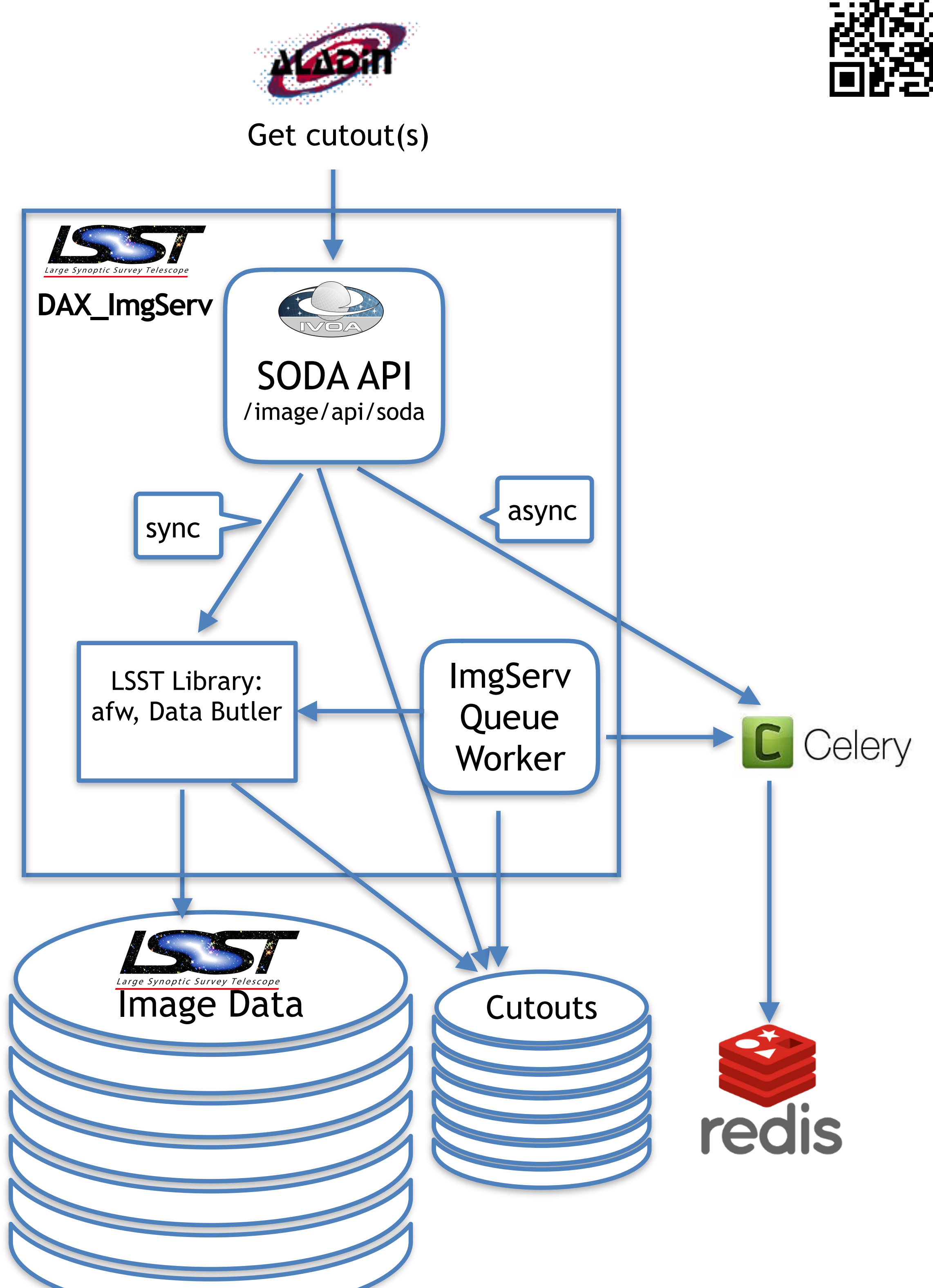
Scalable Image Cutout Service in Python

Kenny Lo *(Data Access)* for the LSST Data Management Team

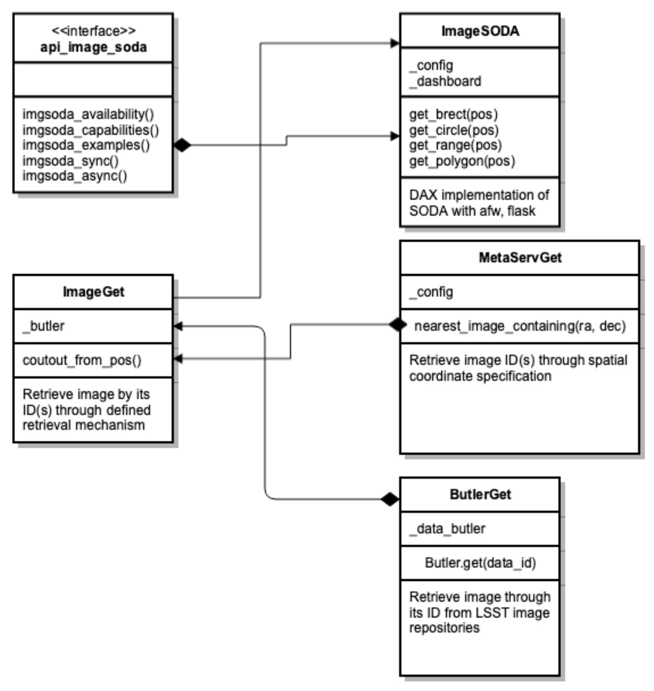
Abstract The Large Synoptic Survey Telescope (LSST) is an 8-m optical ground-based telescope being constructed on Cerro Pachón in Chile. LSST will survey half the sky every few nights in six optical bands. Annual data releases will be made from all the data during the 10-year mission, with unprecedented depth of coadds and time resolution of catalogs for such a large region of sky. Here we present the current status of the image cutout retrieval software based on the LSST software stack to provide API access through the Virtual Observatory (VO) SODA Specification.

The Data Access (DAX) Image Cutout Service, aka DAX_ImgServ

LSST will take about 15 TB of image data per night and after ten years of operations will have 15 petabytes of catalog data from the final data release, and 0.5 exabytes of image data. The LSST camera is designed to provide a wide field of view with better than 0.2 arcsecond sampling and spectral sampling in five or more bands from 400nm to 1060nm. The image surface is flat with a diameter of approximately 64 cm. The detector format will be a circular mosaic providing over 3 Gigapixels per image. DAX_Imgserv enables access for the science users to the massive datasets containing these 6 Gigabyte images at scale.



Top: Architecture Diagram for DAX_ImgServ



DAX_ImgServ Summary

Written in Python 3 as a RESTful *flask* application, using LSST library for data access (aka Data Butler), image cutout and WCS handling. Implemented the VO UWS using celery and Redis for tracking of requests and storing of the results, for asynchronous and batch operations. Both the image data and the results (cutouts) are in distributed storage. For future consideration, sync can be implemented on top of async using URL auto redirection in HTTP.

Left: Main Class Diagram for DAX_ImgServ

Core Components

DAX_ImgServ builds its key components leveraging software packages from well-supported open source projects, and implements the software interfaces defined by the applicable VO Standards.

SODA is LSST implementation of SODA 1.1 in Python 3, which contains the constituent DALI, UWS, and Support Interfaces as defined in their respective VO specification (See More Information below).

daf The Data Access Framework is responsible for mediating between the archive resources and the application writer. The pipeline code has a completely abstract view of file I/O and only has to know how to deal with data objects representing fundamental types such as exposures and tables. Currently FITS is the internal format but the system is designed such that the internal format could be changed to HDF5, for example, and no changes would have to be made to the science pipeline code. This abstraction of the files from the code protects us against shifts in format preferences.

Data Butler of LSST Data Management provides a generic mechanism for persisting and retrieving data using mappers.

afw The Astronomy FrameWork provides the core classes for manipulating exposures and image metadata, including image cutout operations and world coordinate handling.

flask is a micro web framework written in Python.

Celery is an asynchronous task queue/job queue based on distributed message passing.

Redis is an in-memory data structure project implementation a distributed, in-memory key-value database with optional durability.

Service Operation

DAX_Imgserv is deployed in a cluster as a Docker container orchestrated by Kubernetes. For service access, use a SODA client, e.g. web browser or tools e.g. Aladin from CDS. To retrieve a cutout, first query for target the images through VO's Simple Image Access (SIAv1). With the query results, specify the desired dimensions for the cutouts.

SIA Request: /api/image/sia?POS=37.6445, 0.1046&size=0.0278

SIA Response: VOTable in XML

Synchronous (/sync) Requests

Basic service calls:

Get service description: /api/image/soda/sync

Get service availability: /api/image/soda/sync/availability

Get service capabilities: /api/image/soda/sync/capabilities

Basic sync operation:

Get cutout 1: CIRCLE <longitude> <latitude> <radius>
/api/image/soda/sync?ID=X&POS=CIRCLE+37.6445+0.1046+0.0278

Get cutout 2: RANGE <longitude1><longitude2><latitude1><latitude2>
/api/image/soda/sync? ID=X&POS=RANGE+37.6168+37.6723+0.0768+0.1324

Get cutout 3: POLYGON <longitude1><latitude1>...(at least 3 pairs)
/api/image/soda/sync?
ID=X&POS=POLYGON+37.6580+0.0897+37.6580+0.1217+36.6186+0.1006

Responses to /sync requests:

For POS parameter: image files (FITS file)

For all others: XML documents per VO defined Schemas

Asynchronous (/async) Requests

One main attribute of /async request is the presence of unique job_id used on each operation on the request, including tracking the return of the results.

Basic async operation:

Step 1 to start: /api/image/soda/async?
ID=X&POS=CIRCLE+37.6445+0.1046+0.0278
Response: XML per VO Universal Worker Service (UWS)

Step 2 to get job info: /api/image/soda/async/
3605fc07-6193-49b6-824a-90747d1c44fe
Response: XML per UWS

Step 3a for single result: /api/image/soda/async/
3605fc07-6193-49b6-824a-90747d1c44fe/results/result
Response: cutout (FITS format)

Step 3b for multiple results: /api/image/soda/async/
3605fc07-6193-49b6-824a-90747d1c44fe/results
Response: XML per UWS

Step 4 to get job execution duration:
/api/image/soda/3605fc07-6193-49b6-824a-90747d1c44fe/
executionduration

To cancel a pending request:

/api/image/soda/3605fc07-6193-49b6-824a-90747d1c44fe/
phase=ABORT

To get parameters for the job:
/api/image/soda/3605fc07-6193-49b6-824a-90747d1c44fe/
parameters

Obtaining the DAX_ImgServ Software

Latest Docker container image in dockerhub:

```
dock pull webserv/imgserv:dax_latest
```

(requires configuration including /datasets mount to operate)

Source Code: https://github.com/lsst/dax_imgserv

```
git clone https://github.com/lsst/dax_imgserv.git
```

More Information

Aladin: http://aladin.u-strasbg.fr

SODA: http://ivoa.net/documents/SODA

IVOA Documents & Standards: http://ivoa.net/documents

DAX Webservice Implementation Guide: https://dmtn-090.lsst.io

LSST Stack Releases: https://pipelines.lsst.io

LSST Data Products: http://ls.st/LSE-163

DM Applications Design: http://ls.st/LDM-151

Key Numbers: http://lsst.org/scientists/keynumbers

This material is based upon work supported in part by the National Science Foundation through Cooperative Support Agreement (CSA) Award No. AST-1227061 under Governing Cooperative Agreement 1258333 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory. Additional LSST funding comes from private donations, grants to universities, and in-kind support from LSSTC Institutional Members.