Science Platform

The LSST Science Platform (LSP) is the environment provided for user access to and analysis of LSST and user-generated data products.

It presents three "Aspects" to the user:

- The Portal Aspect: a Web application that provides user-friendly, interactive query and visualization tools for catalog and image data;
- The Notebook Aspect: a JupyterLab-based, personalizable environment for interactive Python coding, with access to the LSST data products and additional user computing and storage resources; and
- The API Aspect: a set of Web-based APIs, primarily following IVOA standards, that provide access to the LSST and user-generated catalog and image data products.

These rely on an underlying infrastructure that provides:

- User computing, including resources supporting parallel, next-to-the data analysis;
- A "User Workspace" comprising both a file-oriented storage system and a user database system, each accessible from all three Aspects;
- Database services (including both Qserv and conventional RDBMS components);
- Pre-built releases of the LSST Science Pipelines software stack and other commonly-used Python software, such as Astropy; and
- Authentication and authorization and other support services.

Users can use the Portal Aspect to do queries and visualizations, and UI-driven interactive exploratory data analysis; or they can work in the Notebook Aspect to do Python-based ad hoc data retrieval, analysis, and visualization, using tools of the users choice. Whether from a notebook at the Data Access Center or remotely from an external personal or institutional system, they can also use the Web interfaces of the API Aspect for data access, whether programmatic or using external IVOA-compatible tools such as TOPCAT.

All three of the Aspects are, in effect, front-ends on the databases and files that ultimately will reside in the Data Backbone, and which are shared across Aspects. In the short run, the databases and files live in non-Data Backbone systems.

The system is designed to facilitate analysis workflows that cross from Aspect to Aspect. For instance, the results of a query performed in the UI-driven Portal Aspect can be accessed in the Notebook Aspect via a combination of a simple UI action and Python function invocation; data created from a user analysis in the Notebook Aspect can be visualized in the Portal Aspect, and so on.

We will simultaneously operate multiple distinct instances of this Science Platform. Each instance has its own list of authorized users, update cadence, and upgrade procedures.

Currently (2019) we operate two instances on hardware located at NCSA: a "stable" instance aimed at providing services both internally and to a set of "friendly" users, e.g., from the commissioning team and from Science Collaborations, who wish to familiarize themselves with the Science Platform environment as it continues to develop; and an "integration" instance that provides a platform for testing of new features and fixes for each Aspect in an environment in which the others are available, and that is periodically used for end-to-end tests of the entire Science Platform environment.

Moving into the Commissioning and Operations phases of the project, these instances will be augmented by additional ones:

1. Chilean Data Access Center for science users for released data products.
2. US Data Access Center for science users for released data products.
3. Internal QA of L1 and L2 productions in the production environment. This instance has access to the published Data Backbone contents at the Archive but also has specialized access to unreleased intermediate data products and the internal, unreleased, incrementally-loaded Qserv instance for the next Data Release. In Operations, this instance primarily supports Science Ops. It can also be used by the Commissioning Team. It might have customized portal pages or other components not normally provided in the DAC instances above.
4. Commissioning Cluster at the Base with low-latency access to the Data Backbone endpoint there. This instance primarily supports the Commissioning Team. Any customizations for the QA instance should be available here as well.

The integration instance, at least, will continue to exist indefinitely to support pre-rollout final testing of updates to any of the Aspects or other LSP components.

There may be operational models and requirement relaxations under which some or even all of these instances could be combined.

Initial deliveries of the platform use simple, less-functional components. Later upgrades will improve the components. The initial delivery of (a) a basic Portal integrated with prototype API Aspect services, and (b) a minimally functional notebook-mode QA instance is targeted for some time in Calendar 2017. The delivery of an initial fully integrated version of the other capabilities is targeted for November 2019 in order to precede the start of obtaining on-sky data with ComCam.

Needs Updating

The remainder of this page needs to be updated to be consistent with recent developments, LSE-319, and LDM-542, and in particular to reflect the move of the design to a "VO First" architecture organized around IVOA services.

Portal Aspect:

The initial version of this is the PDAC.

Initial components include:

- SUIT query/visualization portal using Firefly — data retrieved via DAX web services

- DAX web services
    - Low-level:
        - dbserv
            - Raw ADQL/SQL interface, output format translation
            - Talks to Qserv
    - Higher-level
        - metaserv
            - Queries databases (e.g. ScienceCcdExposure table)
            - Generates lists of Butler ids (dataset type plus dataId)
        - imgserv - mosaic/cutout, regeneration, output format translation operations
- Files in GPFS with organization as prescribed in RFC-95, RFC-249
- Qserv database

Later:

An authentication/authorization component will be added that connects to or passes credentials through/to all other components.

A Global Metadata Service will be created to track groups of datasets (Butler repositories) in the Data Backbone. The Global Metadata Service also stores information about available databases.

metaserv then talks to the Global Metadata Service.

imgserv could be expanded to become a read-only "butlerserv". There are two additional functions: returning Butler locations of datasets, which requires a Butler client on the remote end to retrieve and deserialize the datasets, and format translation in which an internal-to-imgserv Butler retrieves the in-memory object for the dataset and streams it to the recipient in a desired format.

Qserv per-user databases will be added as the results of and inputs to portal queries; dbserv will be able to create and query these.

Other RDBMS-based (non-Qserv) databases will be added, including the SQuaSH QC database, provenance databases, and non-Qserv per-user databases; dbserv will be able to create (where appropriate) and query these.

Per-user file storage will also be added.

The Data Backbone will manage the files, replacing the direct GPFS interface (GPFS will still be used underneath). It will perform inter-site replication and transparent (except for latency) retrieval of files from the tape archive. The Butler must be able to retrieve files from the Data Backbone. This can be a staged process (requesting files through a translation dbbToButler utility) and then using a Butler configured to talk to the local filesystem, but it will be more convenient and desirable to have the Butler talk directly to the Data Backbone.

Notebook Aspect:

The initial version of this is for Science Pipelines QA on processed HSC data and does not access SDSS or WISE data in the PDAC.

- Minimal authentication/authorization (Unix user ids on JupyterHub server)
- Local JupyterHub server
- Files in GPFS
- "Monolithic" non-Qserv RDBMS (expected to be MySQL, could even be Oracle or Postgres) instance on new lsst-db containing HSC catalog data products and per-user databases
- Filesystem Butler interface
    - Used with local filesystem and GPFS
- SQLAlchemy (as our current RDBMS-agnostic interface) or Python DB-API interfaces to databases
    - Connects to RDBMS
    - Connects to SQuaSH QC database
- Science Pipelines stack installed and available in the noteboook
- Firefly visualization widgets available in the notebook
- Batch computing on the Verification Cluster via separate shell or shell escape from the notebook

Later:

The Data Backbone and its Butler interface are described above.

DAX services will be implemented to allow added operations on top of file retrieval and database query, including TAP, SIA, and other VO interfaces.

An OpenStack cluster with (for example) Kubernetes is provided for interactive computing.

The JupyterHub server is expanded with features such as:

- Subdomain-per-user and wildcard DNS/HTTPS for security (I think this is best practice)
- KubeSpawner (for example) to provide elasticity for notebooks and compute

The batch cluster could be moved to OpenStack as well.

Straightforward transport of computations from the notebook world to the batch world, controlled by the notebook, remains to be defined.

When Qserv-based data products, per-user Qserv databases, and other RDBMS-based databases are available, connectivity to them through Python DB-API, SQLAlchemy, and the Butler will be provided.