DRP Butler Requirements / Feature Requests

Improved support for skymap-derived data IDs

Jim Bosch:

The skymap packages approach to tessellating the sky is sufficiently flexible that it can be assumed to be the organizational structure for all coadd-based (and likely future multifit-based) data products. The Butler should support automatic iteration over patches given a tract, just as it supports iteration over e.g. CCDs given a visit, as well as iteration over filters.

Unknown User (npease): ⚠ *DM-4181 - Jira project doesn't exist or you don't have permission to view it.*

We also need support for queries that spatially relate skymap-based data IDs to raw data (i.e. visit/ccd) IDs:

- Given a some combination of tract/patch/filter data IDs, return the overlapping raw data IDs.
- Given a some combination of raw data IDs, return the overlapping tract/patch/filter data IDs.

Unknown User (npease): Serge is working on part of this in ⚠ *DM-3472 - Jira project doesn't exist or you don't have permission to view it.* *, but as written it only supports tract/patch/filter to raw dataId. TODO write a story for the other direction.*

Given that the location of a raw dataset is not generally known (or at least not known well) until it is first processed, and it may be improved in subsequent processing, we need to have a mechanism for updating these locations after the data is initially ingested into the data repository.

Unknown User (npease) right now the plan is that locations can be updated/improved by re-running the ingest script described in

⚠ *DM-3472 - Jira project doesn't exist or you don't have permission to view it.*

(A requirement for coadd building; we can hack around it but would urgently prefer not to have to.)

Support for storing, versioning & querying reference catalogs

To replace the existing system of EUPS versioning and astrometry_net_data.

NateP:

⚠ *DM-4168 - Jira project doesn't exist or you don't have permission to view it.* and

⚠ RFC-95 - Jira project doesn't exist or you don't have permission to view it. "Verification datasets: filesystem organization and argument parser features" (search for 'version' and read from there)

Dynamic/generic dataset definition

Jim:

When a new processing step (i.e. top-level Task) is implemented, and this produces on or more output datasets, we need to be able to define those datasets without editing camera-specific files. Ideally we'd be able to define these programmatically at module-import time.

We may also want to provide a template system that allows generic datasets to be defined with camera-specific locations, using predefined locations that can be customized by camera-level mappers. For example, a generic dataset that represents a single-frame processing output could refer to a camera-defined directory for such outputs that includes the data ID.

Unknown User (npease)

- ⚠ DM-4170 - Jira project doesn't exist or you don't have permission to view it. in-repository policy solves some of this issue
- ⚠ DM-4180 - Jira project doesn't exist or you don't have permission to view it. describes the rest, although it's probably going to evolve into more than 1 story.

(No drop-dead date; an issue for L3 processing.)

Integer-Dict Mappings for Data IDs

Jim:

In many contexts we need to transform a dictionary data ID into a unique integer or the reverse. We need a general interface for doing both of these operations (currently only the dict->int transform is implemented, and it's rather ad-hoc and fragile, especially for skymap-based data IDs). This also needs to include a way to calculate the number of bits required for the integer form of the ID. The implementation of these transformations should be delegated to the skymap library for skymap-based data IDs, and delegated to the camera mapper for raw-like data IDs (though a default implementation may be provided by a visit+sensor mapper base class).

(We are accumulating debt the longer we don't have this.)

RHL: the transformation from object ID to e.g. visit ccd objId or tract patch filter objId as appropriate needs to work on numpy arrays. Having written these things, I think the best return type is a dict of numpy arrays. Presumably the current butler's dislike for numpy integer types (actually mysql's dislike) will have been fixed by then.

Unknown User (npease): ⚠ DM-4182 - Jira project doesn't exist or you don't have permission to view it.

Composite Datasets and Caching

See Trac page. Load subsets of larger datasets.

Unknown User (npease): created ⚠ DM-4519 - Jira project doesn't exist or you don't have permission to view it. for this. It will need a lot of fleshing out & definition.

(This already affects performance; the sooner it can be resolved the better.)

Dataset-Level Rerun Granularity

Common source of HSC side bug reports. Discussed by Jim, RHL, K-T, Gregory at DMLT; suggest they present this at Butler meeting.

Unknown User (npease): created for this:

- ⚠ *DM-4520 - Jira project doesn't exist or you don't have permission to view it.*
- ⚠ *DM-4521 - Jira project doesn't exist or you don't have permission to view it.*

Generic visit+sensor data IDs for most cameras

Jim:

Most cameras (the only exception I can think of is SDSS) have a focal plane layout in which each raw data unit can be completely identified merely by specifying an integer for the visit and an integer for the sensor number, and the visit number fully specifies the filter (via registry lookup). This may not be the preferred way of referring to a particular data unit for that camera (e.g. LSST may use rafts, and for some data products also has snaps; other cameras may prefer "ccd" over "sensor"), but if these standard keys were also available for all cameras that could support them it'd be much easier to write camera-independent code. This may require a hierarchy of mapper base classes, with most cameras inheriting from a common base class one level below the top of the hierarchy; algorithms that assume a visit/sensor layout would only be valid on mappers that inherit from that intermediate base class.

RHL: We decided long ago not to require a pre-defined set of names for the items of the data Id (ccd, ccdnum, sensor) and I hope that we're not moving away from this. I understand the desire to make "raft" optional, and I support this. One useful facility would be for a mapper to define an alias e.g. from ccdnum to ccd (an example that the DecamMapper would use).

(Likely a requirement for camera-generic relative astrometry; UW to set schedule?)

Safe concurrent compare/write for provenance files

Should be able to handle writing & checking configurations, schemas and (HSC-side) set up packages, etc, when running parallelized.

Unknown User (npease): ⚠ *DM-4173 - Jira project doesn't exist or you don't have permission to view it.* *takes care of this (right?)*

(Highest priority items are above the line)

Versioning for CameraGeom

Schedule & requirements probably driven by UW. Consider also inheritance-like relationships between cameras which are similar but not identical.

Unknown User (npease) ⚠ *DM-4168 - Jira project doesn't exist or you don't have permission to view it.*

Joint processing from multiple instruments

Requirement for joint processing with WFIRST, etc.

Usable messages/tracebacks from lookup errors

Accumulating debt.

Utility functions for transferring or bundling datasets

Generally useful functionality.

Support relocatable repositories/reruns

Incl. the root of the entire rerun tree.

Support for restarting failed or aborted pipelines

Configuration options for "do I reprocess this thing that has already been processed"; has implications for provenance etc, since whether to reprocess something likely depends on what's been done before.

This is likely an issue for pipeline execution, but may have impacts on the Butler.

(No drop date, but we keep accumulating debt.)

Make it difficult to accidentally clobber somebody else's rerun

Paul/SteveB suggest a "ReadOnlyButler", for e.g.

Better support for different flavors of coadds

Deep, bestSeeing; currently these are done as different datasets, whereas it might be better to make these part of the dataId; coadds with different dates (e.g., nightly coadds), and maybe chi^2 (or multi-band) coadds.

The butler (and all its components) needs to be picklable

So it can be sent (and/or a dataRef) by MPI to worker nodes. It's very convenient to be able to tell a worker to call someTask.run(dataRef). This is dependent on current process middleware; it may not be required in future if we have a better way to transfer data between different threads/processes.

bash command-line completion for --id arguments

Checksums

Convenient way to verify that different repositories really contain the same data.

Network transparency

Need support for reading data on a remote machine with the butler. I don't think we can just use an external solution like sshfs, as sometimes (e.g., HSC data releases) data repos are hosted over http. I want to be able to read inputs from the remote data repo (either http or ssh), cache the inputs in my local data repo (need checksums!), and write outputs on my local data repo. This will be especially useful as LSST moves into production --- for debugging, we will want to be able to inspect and process on our local cluster data products produced at NCSA or IN2P3

Alternative registry databases

Want support for using different databases (e.g., postgres, mysql) for the registries (HSC uses postgres in some places; LSST won't always want to use sqlite: the HSC sqlite registry is nearing 200 MB, and I suspect the entire file is being read every time the butler is instantiated).

Incremental Butler updates

Tell Butler to update itself as more data appears.

Need a means for selecting different types of calibs

So we can try different flats that are valid for the same day. Maybe this is just "use a different calib directory", but then we need to be able to easily use the biases from one calib directory and the flats from another.