LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

# As-is HSC Reprocessing

**Hsin-Fang Chiang**

**DMTN-088**

**Latest Revision: 2018-07-12**

## Abstract

Abstract text.

# Change Record

| Version | Date | Description | Owner name |
|---|---|---|---|
| 1 | YYY-MM-DD | Unreleased. | Hsin-Fang Chiang |

# Contents

# As-is HSC Reprocessing

## 1 Goals

The main goal of the HSC reprocessing is to generate a validation dataset for tests and integration, both scientifically and operationally, and to regularly scrutinize for any data or science issues. Besides allowing to track any quality or performance changes as the pipeline evolves, the processed dataset serves as a base set of data products with which the developers can test a specific component of the science pipelines without the need to reprocess from raw files, or to test a new tool using state-of-the-art data products.

Additionally, the reprocessing helps to stress and validate the infrastructure (both software and hardware) with current science payload, to provide feedback on operational feasibility of all current aspects of the system in use, to explore possible operational strategies, and to mature the policies and procedures of the Batch Processing Service.

Currently, the HSC reprocessing campaigns are performed at two scales: (1) the RC scale, and (2) the PDR1 scale. In general, campaigns are characterized by their goals, inputs (data, calibrations, codes, and configurations), and cadence.

## 2 RC-scale reprocessing

The input data are the HSC “RC2” dataset, which includes 3 tracts of public data and was selected to cover a wide range of data quality and observing conditions (DM-11345). It contains about 8% of the full PDR1 dataset.

A calendar-based schedule is used to run a “mini-DRP” with this RC2 dataset once every two weeks.This repeated reprocessing allows continuous testing and validation of Data Release Production algorithms.

Even-number weekly stack releases, provided automatically by the LSST DM Build system and installed by John in the designated software area on the LSST GPFS, are utilized. To prepare each campaign, 1) verify the software release was successfully built and installed on GPFS, 2) verify ci_hsc can run successfully, confirm resource availability on LSST batch cluster and

storage. Unless otherwise agreed between the DRP and LDF teams before launching the campaign, the default algorithmic configurations are used for the pipelines, as specified in the config files or source codes in the software stack.

Currently a mini-DRP workflow includes the following pipelines: makeSkyMap.py, singleFrameDriver.py, mosaic.py, skyCorrection.py, coaddDriver.py, and multiBandDriver.py. Most frontend codes exist in the pipe_drivers package and are based on the ctrl_pool-style framework. We intend to move away from the ctrl_pool-style framework and work towards a production system; however, the transition is blocked by the Gen3 Butler/Middleware delivery and conversion of pipelines to the new middleware framework.

To monitor and verify if runs are successful, the job status and output data products are checked. Failures are characterized and handled as described in the "As-is mitigation procedures in irregular scenarios" session below.

Generated data products include single frame processing products, coaddition products, and multiband products, and, upon successful completion of a campaign, are made available through the LSST developer infrastructure at NCSA. Management of derived datasets is done at the X level; individuals files are not managed, as would be the case in a production system in which records of location, metadata and provenance are used to centrally manage data. Results of the four most recent successful runs are retained. Datasets are disposed of after they are (prior to being?) superseded by the next campaign.

Currently, though temporarily, the QC/QA prototype pipelines from the pipe_analysis and validate_drp packages are included in the biweekly runs. The pipe_analysis package is not in the official stack of lsst_distrib and was not designed to be included in the official stack in the first place; however, its outputs are essential for pipeline development and QA work (DMTN-074) and no replacement is available yet at time of writing. Neither pipe_analysis nor validate_drp follow the standard pipe_base CmdLineTask framework or the standard Butler usage for data IO.

How the these QC/QA packages will evolve in the long term is a subject of discussions in DMLT and the QA working group. When the DM-wide QA plans are clarified, we will run only the QC/QA packages blessed officially. Before such a plan exists and the packages are standardized, I will continue to work closely with Lauren MacArthur (pipe_analysis) and Michael Wood-Vasey (validate_drp), and Angelo Fausti (dispatch_verify for uploading metrics to SQuaSH) for

emergent issues. As QC output products are not as well-defined, I do not track down outputs as diligently as the official products, and consider the jobs done as long as no fatal errors appear. In the short term, adding automatic testa by running pipe_analysis in the ci_hsc package can reduce obvious breakages.

# 3 PDR1-scale reprocessing

On a per request basis and as deemed necessary, LDF reprocesses the full HSC PDR1 (Public Data Release 1) dataset with a up-to-date DRP workflow and a recent software stack. The reprocessing campaign happens approximately annually. This dataset includes 5654 raw visits in 7 bands and 3 layers (WIDE, DEEP, UDEEP). This covers 119 tracts in the sky.

With the input $\tilde{1}3$ times larger than the RC2 dataset, the PDR1-scale reprocessing helps identifying edge cases and bugs that do not appear in the small dataset. Developers needing more than a few tracts of data for testing algorithms also use the large processed dataset. The processed dataset is also potentially useful for PDAC testing, EPO development, or science verification activities by the commissioning team.

Software version, configurations, and other processing details that affect the scientific output products are decided between the DRP and LDF teams before the start of the run. A RC-scale run with the same setup precedes the PDR1-scale run to integrate and verify the components and configurations are nominally ready for bulk processing. Outputs of the RC-scale run allows QA work and important bug fixes before effort is spent in the large campaign. Typically, a few iterations are involved before the software is finalized and “frozen” for the processing campaign.

Procedures for monitoring and verifying these larger campaigns are similar to the biweekly campaigns. During the campaign, some computing and storage resources are reserved for the campaign use. Completion of these larger campaigns is announced on the LSST Community forum for broader consumption beyond the DM team. The output data products are protected against changes or disasters. Results of the two most recent successful runs are retained. Old data products are disposed before the third campaign starts.

# 4 As-is mitigation procedures in irregular scenarios

Sometimes, things don't go as smoothly as hoped. Here I describe some common scenarios and the as-is procedures for handling the problems.

Weekly release is not available in the shared stack at /software: wait patiently If a weekly stack is not released or tagged in eups, I wait quietly; usually Josh Hoblitt (SQuaRE) would be already aware and working on it. If the weekly release has been eups-tagged successfully but the installation in the /software shared stack fails, I bug John Swinbank. (p.s. we are investigating to switch to using Singularity container for the science payload; this would break our dependency on John's installation.)

- **Failures in building the non-official packages: file a JIRA ticket**

  Until 2018-03-31, two packages, meas_mosaic and pipe_analysis, needed to be updated and built manually. If they cannot be built, for example because the unit tests of meas_mosaic failed, a ticket will be filed so the DRP team can fix it. Meanwhile I start the first processing step which does not need meas_mosaic. Since w_2018_13, meas_mosaic has been added to be part of the lsst_distrib weekly release. So only one package, pipe_analysis, needs special care.

- **Operator user errors: operator's responsibility**

  The DRP workflow in the current system implies a science workflow and there are implicit dependencies in job execution. Currently, this level of workflow is taken care of by the operator manually. If the implicit workflow is not respected, necessary inputs of a job may not be available, so the output products would not be produced correctly, possibly without clear errors in the execution logs. It's the operator's responsibility to shepherd the workflow. Other operator errors include specifying output location improperly, overwriting files incorrectly, operator-introduced race condition, and so on. Some operations require knowledge in Butler and task framework implementations so misunderstanding of the framework can lead to mistakes as well. Many of these errors can be minimized once a real production system, including file management, is in use. The pipeline specification definition is also a design goal of the Gen3 Middleware.

- **Transient failures of the processing jobs due to hardware issues: LDF handles it**

  Hardware or network glitches such as temporary file system unavailability or a faulty node can fail jobs. The operator contacts the infrastructure team and creates IHS tick-

ets as needed. Once the system is back to stable, the same job will be re-submitted. Choosing the operational setup wisely is operator's responsibility. For example, inferior specifications in computing resource need can lead to insufficient memory, timeout, job lingering forever, or other problems.

- **Reproducible fatal errors from the pipeline: report, apply fix if available quickly**

  Occasionally, the weekly stack is broken and does not allow the job to finish. I will verify if the failure is reproducible, and report to the Slack channel #dm-hsc-reprocessing for the DRP team to respond. Sometimes the bug is obvious, such as one from recent API changes or new features not caught by the standard CI tools; in this case a fix can be available quickly especially with the help of the developer who added the new feature. If a fix is available within a day, I will use the fix and re-do the execution. If the fix is not available shortly, I may pause the campaign (although this hasn't happened so far; developers are always helpful.)

- **Reproducible non-fatal errors from the pipeline: carry on without changes**

  Errors that are not in the FATAL-level do not stop the execution. Processing continue with the same software. Currently some harmless errors appear in the logs and not all errors are carefully vetted. Only severe errors that prevent further execution of the pipelines stop a processing campaign, despite the outputs may or may not be scientifically useful. In the future, the formal QC pipelines will be helpful to provide scientific metrics of the data products during the campaign execution, and may provide additional feedbacks on whether a campaign needs to be paused or specific inputs need to be removed.

## References