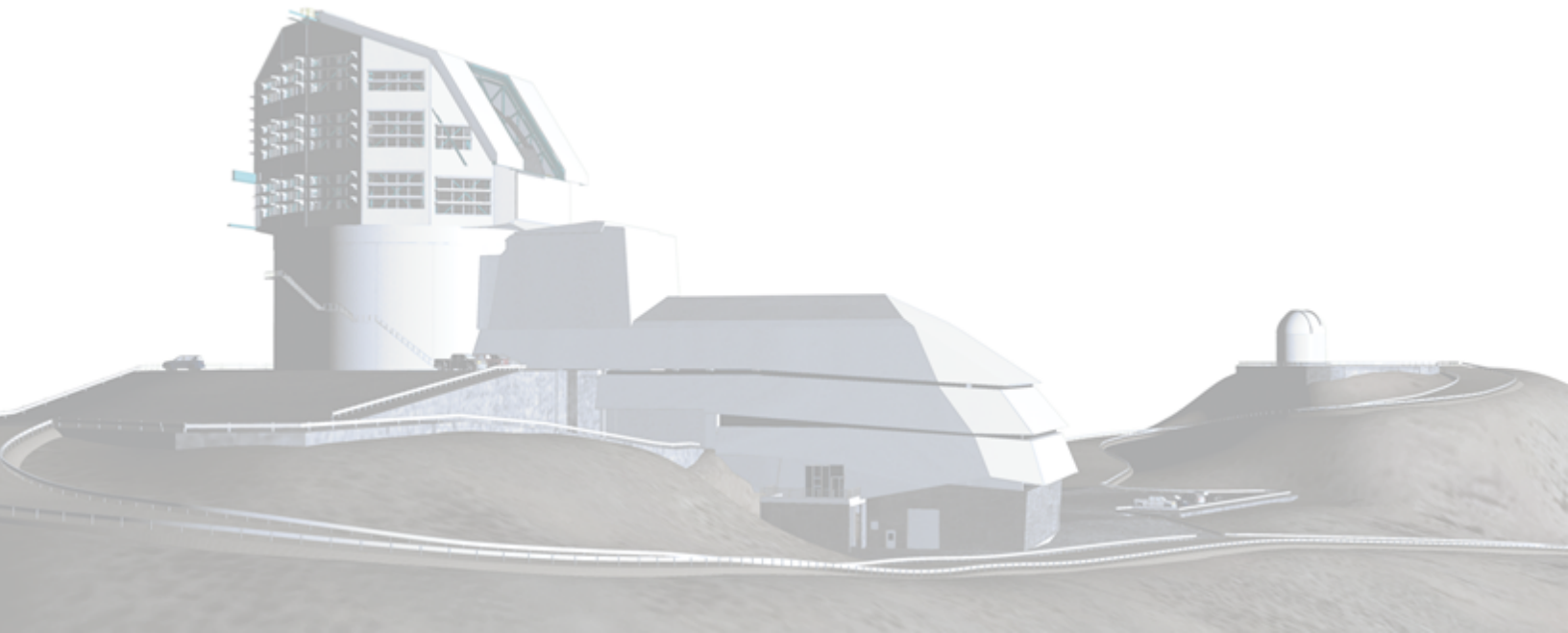

Vera C. Rubin Observatory
Data Management

Ingesting reprocessed HSC catalog data to Qserv at NCSA

Hsin-Fang Chiang

DMTN-170

Latest Revision: 2021-02-05

Abstract

This DMTN describes the ingestion of reprocessed HSC object tables into Qserv instances at NCSA.

Change Record

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

Document source location: `https://github.com/lsst-dm/dmtn-170`

Contents

Ingesting reprocessed HSC catalog data to Qserv at NCSA

This is a report for DM-22806 on automating the process to take LSST science pipelines outputs, produce parquet files, conform to the Science Data Model (SDM) format, and ingest them into Qserv. The Qserv instance is deployed at NCSA accessible directly for LSST developers and via Science Platform for users.

The User Guide of the Qserv Ingest System is at `https://confluence.lsstcorp.org/x/WoPyBw`, where one can find detailed documentations on the Qserv Ingest APIs, example usages of the Ingest System, example workflows, and advanced ingest scenarios. This DMTN does not intend to duplicate the User Guide; instead, this DMTN focuses on the end-to-end workflow starting with LSST science pipelines outputs and the current prototype system as implemented specifically for the NCSA environments.

1 Input data from Science Pipelines

The input data are the reprocessed HSC products available at Rubin's GPFS space `/datasets/hsc/repo/rerun/`. The object tables produced by the postprocessing pipeline, part of the DRP processing, are used. These object tables, with the Butler dataset type of `objectTable_tract`, are provided one file per tract in the parquet format on the shared filesystem. In the case of the HSC-RC2 dataset, there are three tracts in total, hence three parquet files are provided in each reprocessing rerun.

2 Overall workflow

Figure 1 illustrates the overall workflow from the input parquet files to an ingested database in the Qserv instance.

Constrained by the workflow management systems provided to DM developers on the Rubin Batch Systems [1], the ingest workflow is implemented with Pegasus [2] for running on the HTCondor DAC Cluster. The workflow code is at `https://github.com/lsst-dm/qserv-ingest-hsc-poc`. The following sessions dive into individual steps of the workflow.

[1] `https://developer.lsst.io/services/batch.html`
[2] `http://pegasus.isi.edu/`

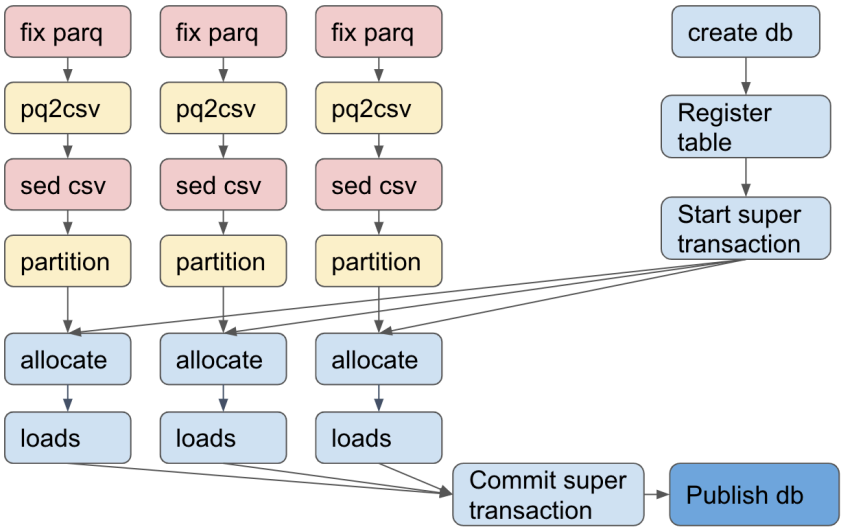

FIGURE 1: Workflow Diagram. Blue boxes indicate direct connections with the Qserv cluster; red boxes indicate temporary workarounds.

3 Conforming parquet files to SDM

The schema definitions of the Rubin Science Data Model (SDM) are felis-[3]-style YAML files and stored in the `sdm_schemas` package [4]. The package is versioned with the `lsst_distrib` stack. In `ci_hsc_gen2`, the column names of the pipeline outputs are verified against the `sdm_schemas/yml/hsc.yml` file. As of this writing, the data types in the parquet files do not perfectly match SDM yet. One issue is null handling. While the convention to handle missing data is being discussed (DM-25926) and the parquet files do not have SDM types, a step is added in the workflow to replace all nulls with 0 and to set column types following `sdm_schemas/yml/hsc.yml`.

4 Converting parquet to csv

The `parquet_tools` package [5] is used to convert parquet files into csv format. `pq2csv` expects a schema file and also checks the schema consistency with the input parquet files. Currently `pq2csv` expect this schema in its own customized format, not felis.

[3] `https://github.com/lsst-dm/felis`
[4] `https://github.com/lsst/sdm_schemas`
[5] `https://github.com/lsst-dm/parquet_tools`

5 Fixing csv

Boolean values should be represented by literals, so `Ture` and `False` are replaced by `1` and `0`. Currently, `inf` and `-inf` are replaced by null values (`\N`). The linux `sed` command is used.

6 Partitioning into chunk files

The spatial data partition is performed by the `partition`[6] package. Output chunk files are directly stored onto the GPFS shared filesystem and are not tracked by the workflow manager.

7 Using Qserv Ingest web services

Qserv Ingest web services have REST APIs and the HTTP requests may be sent directly or via python code. Our ingest client code is written in python utilizing the `requests` library. Most requests expect a JSON object for the parameters. These requests allow users to create a database, create a table, start a super-transaction, allocate chunks, commit a super-transaction, publish a database, and so on. Suitable ports have been opened between the Rubin Batch Systems and the Qserv cluster to allow these HTTP requests.

8 Data loading

There are two ways to load data. Previously, the binary tool `qserv-replica-file-ingest` available via the docker container was used. Due to the need of docker, it cannot be run on the clusters of the Rubin Batch Systems. Therefore, we did this step on Qserv machines on either masters or workers. Recently, a new feature has been added (DM-27091) for the workers to ingest files directly from the mounted filesystem, or by pulling from a remote object store. The Qserv workers now have built-in REST servers to load the data. The new feature allows data loading to be done from the Rubin Batch Systems where docker cannot be used.

No data movement of the chunk files is involved as the chunk files are stored on the GPFS shared filesystem mounted on both the Rubin Batch Systems and the Qserv cluster.

[6] `https://github.com/lsst/partition`

Database Name	Dataset	Row Count	Input Size	Tract Count	Job Count	Wall time	Cumulative time
hsc_pdr2_2020_08_01	HSC-PDR2 DEEP	32193313	45G	39	198	~32m	~5h39m
hsc_pdr2_2020_09_00	HSC-PDR2 WIDE	570287192	659G	663	3318	6h27m	5d9h
hsc_rc2_w_2020_30_04	HSC-RC2	5520644	8G	3	18	26m	1h1m
hsc_rc2_w_2020_38_04	HSC-RC2	5528373	8G	3	18	25m	1h3m
hsc_rc2_w_2020_42_05	HSC-RC2	5516628	8G	3	21	29m	1h13m
hsc_rc2_v21_0_0_rc1_04	HSC-RC2	5525413	8G	3	21	29m	1h13m
hsc_rc2_w_2021_02_00	HSC-RC2	5516315	8G	3	21	27m	1h9m
dc2_object_run22i_dr6_wfd_v1_08	DC2 DR6 WFD	147088445	118G	166	999	47m	11h23m

TABLE 1: An overview of current databases available in the "small" Qserv instance at NCSA. These databases contain Object tables. The job count, wall time, and cumulative job wall time of the ingest workflow for each batabase are also shown. The workflow job counts do not include additional data transfer or management jobs added by the workflow manager Pegasus. The timing information is provided by Pegasus. The ~ symbol indicates missing timing data; estimates are given based on the same loading in the "large" Qserv instance. The input size is the sum of all input parquet files. One DC2 database is included for comparison.

9 Examples

Table 1 lists the databases availabe in the "small" Qserv instance at NCSA as of this writing. New object tables from the periodic HSC-RC2 reprocessing continue to be ingested into Qserv. Some of these databases were ingested before the new data loading feature was implemented and the data loading was done on the Qserv cluster (see Sect 7), hence the job counts differ among the HSC-RC2 databases. `w_2020_42` was the first HSC-RC2 database with data loading directly issued on the Rubin Batch Systems. Some, but not all, of these databases are made queryable via TAP on Rubin Science Platform hosted at NCSA.

A References

B Acronyms

Acronym	Description
DAC	Data Access Center
DC2	Data Challenge 2 (DESC)
DM	Data Management
DMTN	DM Technical Note
DRP	Data Release Production
GPFS	General Parallel File System (now IBM Spectrum Scale)

HSC	Hyper Suprime-Cam
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
NCSA	National Center for Supercomputing Applications
PDR2	Public Data Release 2 (HSC)
REST	REpresentational State Transfer
TAP	Table Access Protocol
WFD	Wide Fast Deep
YAML	Yet Another Markup Language