



Vera C. Rubin Observatory
Data Management

Campaign Tooling – tools for generating, monitoring and tracking data processing campaigns

Brian Yanny, Colin Slater, Sergey Padolski, Kian-Tat Lim, Michelle
Gower, Yusra AlSayyad, Hsin-Fang Chiang, Huan Lin

RTN-023

Latest Revision: 2021-09-23



Abstract

Tools for working with the Butler to query for status of data collections to generate processing campaigns, for working with BPS, PanDA, Condor or other workflow systems to monitor campaigns; tools to track success and failure of campaigns, past and present. Requirements described.

Change Record

Version	Date	Description	Owner name
1	2021-08-09	Initial release for feedback.	Brian Yanny
2	2021-08-17	Addressing comments in pull/2.	Brian Yanny

Document source location: <https://github.com/lst/rtn-023>

Contents

1 Introduction and Definitions	1
2 Campaign Generation and Tracking Requirements	2
3 BPS and the PanDA system	4
3.1 Monitoring jobs in PanDA	6
3.2 More PanDA monitoring	9
4 Sample BPS 'run' specification yaml	13
5 Implementation Discussion Points	15
6 Next steps in Campaign Tooling Development	17
7 DRP.yaml steps	18
A References	22
B Acronyms	23

Campaign Tooling – tools for generating, monitoring and tracking data processing campaigns

1 Introduction and Definitions

Data Processing for Rubin occurs at several levels, from the lowest level ‘quantum’ of processing, to the ‘task’ level for running the same routine on many quanta, to a ‘run’ which is a ordered set of tasks where the order and tasks are nodes in a ‘directed acyclic graph’ (DAG).

This document draws heavily from Lim et al. (DMTN-181) which defines campaigns, and the scope and role of a campaign management system for data processing. Some more related definitions are described in Chiang et al. (DMTN-137). The difference between visit and exposure ids is explained in Jenness (iddefs).

A high-level term to describe processing is ‘campaign’.

A ‘campaign’ is defined as one or more related sets of runs, to carry out a specific data high-level processing goal.

Examples of campaigns are:

- (re)process the DC2 dataset, starting with the raw calibration and visit exposures, into tracts.
- Create flat-field, dark, bias calibrations from raw flat-field, dark, bias exposures. Build master calibrations.
- Process last night’s data from the mountain top (some 1000-2000 visits typically) through the single frame processing steps.
- Perform FGCM calibration on all consolidated visits within a given date range.
- Generate coadds from a set of consolidated visits in a given observing date range that overlap a given set of tracts; use a specific FGCM calibration. The set of consolidated visits may be chosen based on a set of science quality metrics for each visit, which allows cuts on parameters such as seeing < seeing-threshold, sky brightness < sky-threshold, object count > count-threshold, etc.

- Perform Difference Imaging Analysis on a specific set of visits, which may include generation of comparison templates from visits/coadds of a given date range.
- Make a data release (i.e. DR1). A very high level campaign!

It is necessary, when managing large, complex sets of runs in campaigns, such as for nightly or annual Data Release Processing (DRP) to:

- generate parameters for describing related sets of a campaign's runs in a systematic fashion.
- track all runs and their associated control parameters and settings for a campaign.
- monitor run status for all runs within a campaign, and all currently running or recently ended campaigns
- review stdout and stderr and other log files for individual jobs within runs, especially when a job or quanta within a job fails.

Campaign monitoring tools are also potentially closely connected to System Performance metrics such as described in Economou (DMTN-173), for instance Faro checks on pipeline outputs Carlin (Metrics), Bechtol (FARO).

The purpose of this document is to describe requirements on a campaign tracking and monitoring system that can begin by assisting in DP0 O'Mullane (RTN-001) production and grow to assisting in DRP and other data processing for Rubin.

2 Campaign Generation and Tracking Requirements

To generate processing campaigns and track them these features are desired, as shown in Figure 1:

1. Ability to generate BPS submit yaml configs (or other top level workflow submit scripts) which request processing of subset of inputs needing to be processed, pipelines to be

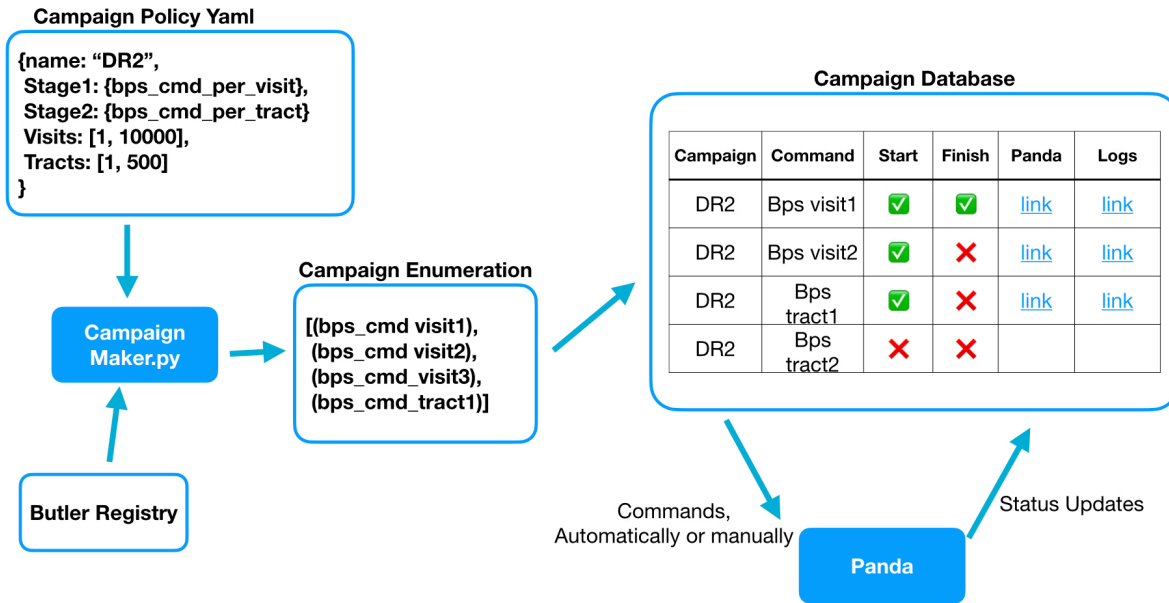


FIGURE 1: Diagram of Campaign generation (left) and monitoring system (right). Credit: C. Slater

run, outputs described. This could, for instance, be achieved by having a set of template yaml configs – one for each processing step (see step list for DRP below) (generate calibs from raw calibs, Single Frame, Consolidate Visit, Photo calib FGCM, astrometric calib skyMap, make coadd for a set of tracts, run photometry on the coadd, run forced Photometry on the individual visits of a coadd, Difference Imaging Analysis) where a 'sed' script or similar would fill in a visit list, a src catalog list, a tract list, etc.

- Should be able to 'split' a large set of inputs into several manageable sized pieces. For instance, DP0.2 consists of about 40,000 visits, and a typical Rubin night of observing will take 1,000-2,000 science exposures. So a manageable unit for single visit processing might be of size 1000 exposures/visits and a DP0.2 processing would split the 40,000 visits into 1000 visit groups. Consideration should be given to how long (wallclock time) a single campaign 'unit' may be run and to not generally exceed 24 hours to a few days to avoid interruptions from downtimes. Also depends on ability of system to recover from interruptions and continue without needing to restart the whole campaign from the beginning.
- Should allow 'last minute editing' of configs before submission to tweak something, change a pipeline version or visit input list. This means that whatever records the

campaign parameters is able to record the change as well and not get out of sync with the by-hand or last minute change.

- Each campaign should be identified by some unique id string, perhaps same as workflow manager identifying string? There is a concept of a RUN which can serve as at least part of an identifying string.

2. Ability to store and manage a set of BPS submit scripts (or equivalent).

While one doesn't wish to reproduce the butler registry, there is a need to store somewhere campaign or run 'tuples' which can be a set of BPS submit file.yaml along with information such as time-of-submission and current status.

3. Ability to quickly access and view logs of each campaign's individual tasks/jobs to determine errors

4. Ability to view overall state of running and recently completed campaigns, without being overwhelmed by long-ago completed or abandoned campaigns.

5. As centrally (DF) submitted runs may be distributed to multiple sites for execution, managers at local sites should be able to view only those jobs running at their local site on local resources.

6. Ability to watch, in realtime, as jobs or tasks within a campaign or set of campaigns are running.

The JIRA system is a candidate place where these campaign-grouped submit file run 'tuples' could be stored. Each campaign would have a JIRA ticket number and could include sub-tickets on each BPS submit run for the campaign. Another option would be a separate (outside or inside the Butler) Database table such as that which might hold an exposure catalog.

3 BPS and the PanDA system

For DP0 O'Mullane & Dubois (RTN-013), Rubin is using the PanDA (PanDA) workload data. The Batch Production Service (BPS) Kowalik et al. (LDM-636) is used as an interface to PanDA, with some BPS use cases described in Kowalik et al. (LDM-633).

BPS:

1. BPS accepts a yaml formatted keyword-value set of parameters which defines a single run. (see example run specification yaml below).
2. After 'bps submit run-spec.yaml' is entered on the command line, BPS launches a butler query to generate a 'quantum graph', a DAG which defines the ordered set of pipetasks which are to be run on which specific set of input collections (or subsets of a collection which may be picked out via SQL-style where clauses).
3. BPS generates for the requested pipeline, an ordered list of jobs, each of which may consist of hundreds of quanta, and submits this list to a workload system (such as HT-Condor or PanDA).
4. The BPS system, in Execution Butler mode (which is the default mode for processing) creates, and after the quantum graph is generated for the run, a small (currently sqlite) database containing all the necessary Butler information (metadata and data storage link URIs) to process all steps in the run. This Execution Butler is constantly updated by the processing the data in question.
5. If data is being processed at a remote site, the bps submit step could also help initiate the moving of centrally stored files to a remote processing data store, using, for instance, Rucio+FTS.
6. BPS monitors runs which have been submitted (in the HT-Condor case see Fig. 2; in the PanDA case, there may need to be more feedback send back, or pointers on how to poll for status).
7. When the processing of all tasks in a run completes, a 'merge' step is run by BPS which syncs the small Execution Butler metadata, include info on any new files generated during processing, back into the main Butler.
8. If data is being processed at a remote site, the merge step could also help initiate the moving of local file back to a central data store, using, for instance, Rucio+FTS.

PanDA:

1. Accepts an ordered list of tasks (from BPS), which specify what pipetasks to run on which inputs.
2. Distributes tasks to worker nodes, which may be widely distributed.
3. Monitors tasks and associated quanta. See Figs. 3,4,5,6

Path: /project/huanlin/dc2/submit/u/huanlin/test_ci_imsim_w19/20210507T160046Z

	UNKNOWN	MISFIT	UNREAD	READY	PENDIN	RUNNIN	DELETE	HELD	SUCCEE	FAILED
Total	0	0	0	0	0	0	0	0	787	0
pipetaskInit	0	0	0	0	0	0	0	0	1	0
isr	0	0	0	0	0	0	0	0	36	0
characterizeImage	0	0	0	0	0	0	0	0	36	0
calibrate	0	0	0	0	0	0	0	0	36	0
consolidateVisitSummary	0	0	0	0	0	0	0	0	30	0
makeWarp	0	0	0	0	0	0	0	0	168	0
writeSourceTable	0	0	0	0	0	0	0	0	36	0
transformSourceTable	0	0	0	0	0	0	0	0	36	0
consolidateSourceTable	0	0	0	0	0	0	0	0	30	0
assembleCoadd	0	0	0	0	0	0	0	0	64	0
detection	0	0	0	0	0	0	0	0	64	0
mergeDetections	0	0	0	0	0	0	0	0	17	0
deblend	0	0	0	0	0	0	0	0	17	0
measure	0	0	0	0	0	0	0	0	64	0
mergeMeasurements	0	0	0	0	0	0	0	0	17	0
forcedPhotCoadd	0	0	0	0	0	0	0	0	64	0
writeObjectTable	0	0	0	0	0	0	0	0	17	0
transformObjectTable	0	0	0	0	0	0	0	0	17	0
forcedPhotCcd	0	0	0	0	0	0	0	0	36	0
consolidateObjectTable	0	0	0	0	0	0	0	0	1	0

####

FIGURE 2: Sample BPS report display showing number of quanta for each task within a give run. Credit: M. Gower

3.1 Monitoring jobs in PanDA

PanDA can accept http:// url queries to access the current status of running and completed jobs.

These URLs may be accessed with a python script. http:// may be used rather than https:// to avoid needed to authenticate credentials.

Warning: There is currently a limit of some 500 requests/hour to avoid getting IP blocked. This can be adjusted if needed.

A term 'json' may be added to the URL to format the output as json text.

Sample syntax for PanDA running/completed job status queries::

http://panda-doma.cern.ch/jobs/?date_from=21-09-2021&date_to=22-09-2021

There are three levels of monitoring: workflows, tasks and then jobs at the lowest level.

One sees (current) workflows with a query such as:

<http://panda-doma.cern.ch/idds/wfprogress>

or in json format:

<http://panda-doma.cern.ch/idds/wfprogress/?json>

(this seems to not quite be in standard json format, but look at the raw output to extract tasknames)

This returns json formatted text which includes "run names" `r_name` (one per BPS submit), one can pick out one or more run names of interest and look up the tasks associated with that run name (note the `*` construct in the URL):

http://panda-doma.cern.ch/tasks/?taskname=u_huanlin_panda_test_ci_imsim_d_2021_09_15_w38_20210923t1

By default the search looks for jobs within the last 7 days, one may search a range of times/dates by specifying a date or datetime string:

https://panda-doma.cern.ch/jobs/?date_from=21-09-2021T10:00&date_to=21-09-2021T11:00

This particular run, which is a submission of a 'step3' (see below) from the DRP processing list, contains 14 'tasks' (as PanDA calls them). These have names:

PREFIX="u_huanlin_panda_test_ci_imsim_d_2021_09_15_w38_20210923t155401z"

```
reqid: 6237 taskname: ${PREFIX}_deblend_3811
reqid: 6236 taskname: ${PREFIX}_healSparsePropertyMaps_3809
reqid: 6235 taskname: ${PREFIX}_transformObjectTable_3816
reqid: 6234 taskname: ${PREFIX}_mergeDetections_3810
reqid: 6233 taskname: ${PREFIX}_selectGoodSeeingVisits_3804
reqid: 6232 taskname: ${PREFIX}_measure_3812
```

```
reqid: 6231 taskname: ${PREFIX}_templateGen_3807
reqid: 6230 taskname: ${PREFIX}_assembleCoadd_3806
reqid: 6229 taskname: ${PREFIX}_makeWarp_3805
reqid: 6228 taskname: ${PREFIX}_pipetaskInit_3803
reqid: 6227 taskname: ${PREFIX}_detection_3808
reqid: 6226 taskname: ${PREFIX}_consolidateObjectTable_3817
reqid: 6225 taskname: ${PREFIX}_forcedPhotCoadd_3814
reqid: 6224 taskname: ${PREFIX}_mergeMeasurements_3813
reqid: 6223 taskname: ${PREFIX}_writeObjectTable_3815
```

Note that the order here of the reqid's doesn't match the order of the tasks in the step3 definition, which is in this file:

```
$OBS_LSST_DIR/pipelines/imsim/DRP.yaml
```

This file defines step3 as these tasks in this order: step3: subset:

1. makeWarp
2. assembleCoadd
3. detection
4. mergeDetections
5. deblend
6. measure
7. mergeMeasurements
8. forcedPhotCoadd
9. transformObjectTable
10. writeObjectTable
11. consolidateObjectTable

12. healSparsePropertyMaps
13. selectGoodSeeingVisits
14. templateGen

Sorting the tasks by the numerical suffix at the end of the task name:

```
reqid = task
```

```
reqid: 6228 taskname: ${PREFIX}_pipetaskInit_3803
reqid: 6233 taskname: ${PREFIX}_selectGoodSeeingVisits_3804
reqid: 6229 taskname: ${PREFIX}_makeWarp_3805
reqid: 6230 taskname: ${PREFIX}_assembleCoadd_3806
reqid: 6231 taskname: ${PREFIX}_templateGen_3807
reqid: 6227 taskname: ${PREFIX}_detection_3808
reqid: 6236 taskname: ${PREFIX}_healSparsePropertyMaps_3809
reqid: 6234 taskname: ${PREFIX}_mergeDetections_3810
reqid: 6237 taskname: ${PREFIX}_deblend_3811
reqid: 6232 taskname: ${PREFIX}_measure_3812
reqid: 6224 taskname: ${PREFIX}_mergeMeasurements_3813
reqid: 6225 taskname: ${PREFIX}_forcedPhotCoadd_3814
reqid: 6223 taskname: ${PREFIX}_writeObjectTable_3815
reqid: 6235 taskname: ${PREFIX}_transformObjectTable_3816
reqid: 6226 taskname: ${PREFIX}_consolidateObjectTable_3817
```

One may lookup status information about a specific task (=reqid) by using this syntax:

For status of the deblend task:

```
https://panda-doma.cern.ch/task/6232/?json
```

3.2 More PanDA monitoring

1. Allows browsing of log files from individual tasks. See Fig 7,8

PanDA Dash Tasks Jobs Errors Users Sites Harvester My BigPanDA Help

Workflow monitor a-ids-03 Refresh

Requests:

Show 10 entries Search: huanlin

request_id	workflow status	workflow name	created on (UTC)	total tasks	tasks	remaining_files	processed_files	total_files
179	finished	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z	2021-08-05 15:38:55	23	Finished(23)	0	4091	4091
159	finished	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_try2_20210730t012146z	2021-07-30 01:25:31	23	Finished(23)	0	15579	15579
158	finished	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_20210729t152013z	2021-07-29 15:22:58	14	Finished(14)	0	2918	2918
157	finished	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_20210729t022506z	2021-07-29 02:27:20	7	Finished(7)	0	5765	5765
154	finished	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_20210728t214929z	2021-07-28 21:54:29	4	Finished(4)	0	6898	6898
134	finished	u_huanlin_panda_test_ci_imsim_w26_try4_20210726t201741z	2021-07-26 20:18:05	23	Finished(23)	0	850	850
127	expired	u_huanlin_panda_test_ci_imsim_w26_try3_20210726t171026z	2021-07-26 17:11:44	31	Failed(31)	1165	1	1166
123	finished	u_huanlin_panda_test_ci_imsim_w26_small_20210722t010139z	2021-07-22 01:01:50	4	Finished(4)	0	4	4
122	finished	u_huanlin_panda_test_ci_imsim_w26_20210722t005825z	2021-07-22 00:59:47	23	Finished(23)	0	850	850
108	subfinished	u_huanlin_panda_test_ci_imsim_w26_small_20210715t171436z	2021-07-15 17:15:53	4	Finished(2) Failed(2)	1	3	4

FIGURE 3: Top level PanDA monitoring dashboard of running and recent jobs Credit: S. Padolski

← → ↺ 🏠 🔍 https://panda-doma.cern.ch/tasks/?taskname=u_huanlin_panda 80% ☆ 🔍 Search ☆ 📄 📄 📄 📄 📄

4822	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_detection_2459 test lsst atpilot1 RequestID: 4822 Errors	done 1	100% 1		2021-08-05 19:18:24	2021-08-05 19:18:24	900		LSST
4820	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_writeSourceTable_2451 test lsst atpilot1 RequestID: 4820 Errors	done 603	100% 603		2021-08-05 19:14:23	2021-08-05 19:14:23	900		LSST
4818	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_assembleCoadd_2455 test lsst atpilot1 RequestID: 4818 Errors	done 1	100% 1		2021-08-05 19:07:22	2021-08-05 19:07:22	900		LSST
4826	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_makeCoaddVisitTable_2457 test lsst atpilot1 RequestID: 4826 Errors	done 1	100% 1		2021-08-05 18:59:21	2021-08-05 18:59:21	900		LSST
4830	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_makeWarp_2452 test lsst atpilot1 RequestID: 4830 Errors	done 153	100% 153		2021-08-05 18:56:07	2021-08-05 18:56:07	900		LSST
4813	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_makeVisitTable_2456 test lsst atpilot1 RequestID: 4813 Errors	done 1	100% 1		2021-08-05 18:55:21	2021-08-05 18:55:21	900		LSST
4815	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_consolidateVisitSummary_2453 test lsst atpilot1 RequestID: 4815 Errors	done 153	100% 153		2021-08-05 18:50:43	2021-08-05 18:50:43	900		LSST
4816	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_calibrate_2450 test lsst atpilot1 RequestID: 4816 Errors	done 603	100% 603		2021-08-05 18:37:35	2021-08-05 18:37:35	900		LSST
4814	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_characterizeImage_2449 test lsst atpilot1 RequestID: 4814 Errors	done 603	100% 603		2021-08-05 18:12:04	2021-08-05 18:12:04	900		LSST
4821	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_isr_2448 test lsst atpilot1 RequestID: 4821 Errors	done 603	100% 603		2021-08-05 16:33:07	2021-08-05 16:33:07	900		LSST
4817	u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805t153638z_pipelineTask_2447 test lsst atpilot1 RequestID: 4817 Errors	done 1	100% 1		2021-08-05 15:52:49	2021-08-05 15:52:49	900		LSST

FIGURE 4: Next level task list for first job in list Credit: H. Lin

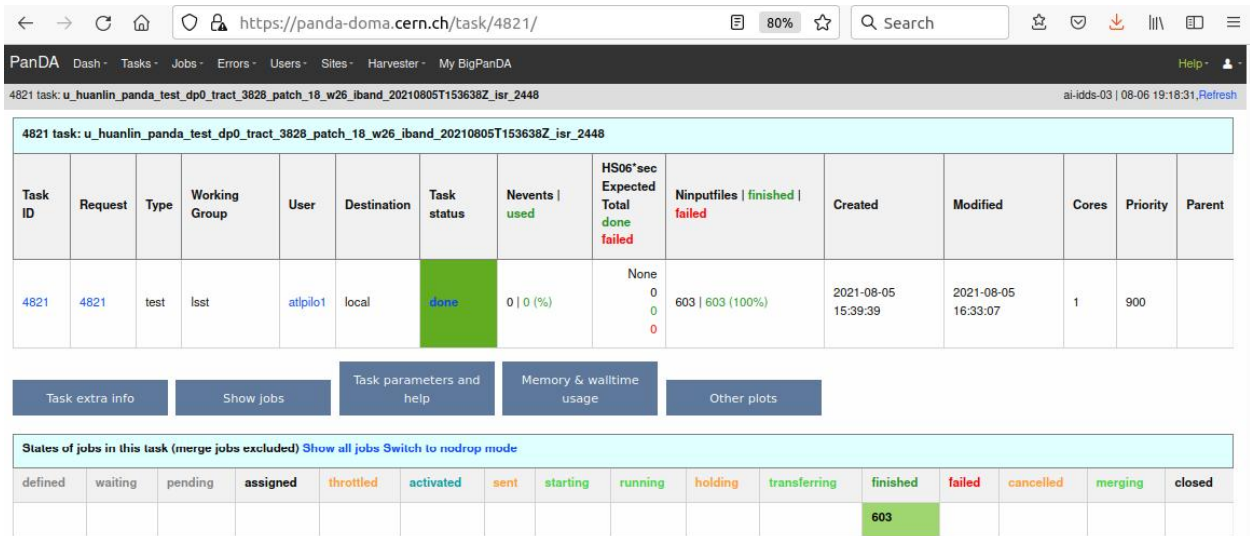


FIGURE 5: Next level status for isr task (2nd from bottom in prev list). Credit: H. Lin



FIGURE 6: Next level status for some of the 603 quanta in this task. Credit: H. Lin

PandaID	Owner WG / VO	Request Task ID	Status	Created	Time to start d:h:m:s	Duration d:h:m:s	Modified	Site	Priority
1449432	atlpilo1 / lsst / wicg	4821 4821	finished	2021-08-05 16:01:14	0:0:23:40	0:0:01:05	2021-08-05 16:26:07	DOMA_LSST_GOOGLE_TEST	900
Job name: u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805T153638Z_isr_2448.1449432 transformation: bash-c-enc									
Datasets: Out: PandaJob_#(pandaid)/									
Dataset summary: log: 1; pseudo_input: 1									

Logs	Go to	Show	Jump to	Memory and IO plots
<ul style="list-style-type: none"> Pilot stdout Job stderr job stdout Pilot job jdl Pilot records Antionn logger (Kihana) Open all logs log.gz (log) Rucio DDM Dash (Grafana) 				

Scope	Size (MB)	Status	Attempt (max)	Dataset
u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805T153638Z.qgraph+384_isr_994987_28+1628177849.2402253-7002+384	0.00	unknown		pseudo_dataset Rucio
PandaJob_#(pandaid)/	0.01	ready		PandaJob_#(pandaid)/ Rucio (destination block: #(pandaid)/ Rucio)

FIGURE 7: Getting to the stdout/stderr log for this quantum. Credit: H. Lin

```

← → ↺ 🏠 🔒 https://storage.googleapis.com/drpf-us-central1-logging/logs/DOMA_LSST_GOOGLE_TEST/1449432/job/1449432.log
🔍 Search 📌 📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📘 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿

anaconda-post.log
bin
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
Creating user data directory: /tmp/.eups
/opt/lsst/software/stack/miniconda3-py38_4.9.2-0.6.0/Linux64/ctrl_mpxec/21.0.0-34-g2cbe45c+03660fcc69/bin/pipetask run -b s3://butler-us-central1-panda-dev/butler.yaml -i 2.21/defaults/DP0.2 --output u_huanlin/panda_test_dp0_tract_3828_patch_18_w26_iband --output-run u_huanlin/panda_test_dp0_tract_3828_patch_18_w26_iband/20210805T153638Z --extend-run --skip-init-writes --qgraph s3://butler-us-central1-panda-dev/hsc/payload/u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805T153638Z/u_huanlin_panda_test_dp0_tract_3828_patch_18_w26_iband_20210805T153638Z.qgraph --qgraph-id 1628177849.2402253-7002 --qgraph-node-id 384 --clobber-outputs --skip-existing --no-versions
botocore.credentials INFO: Found credentials in environment variables.
matplotlib.font_manager INFO: generated new fontManager
ctrl_mpxec.cmdLineFwk INFO: QuantumGraph contains 1 quanta for 1 tasks, graph ID: '1628177849.2402253-7002'
isr INFO: Converting exposure to floating point values.
isr INFO: Assembling CCD from amplifiers.
isr INFO: Applying bias correction.
isr INFO: Applying crosstalk correction.
isr.crosstalk INFO: Applying crosstalk correction.
isr INFO: Masking non-finite (NaN, inf) value pixels.
isr INFO: Widening saturation trails.

```

FIGURE 8: A look at stdout/stderr – if there was an error, one can view it here to help diagnose an issue. Credit: H. Lin

2. Allows for retries with more memory for individual tasks that fail.
3. There are graphana, kibana elasticsearch plots that can be setup to monitor overall PanDA system status and performance.

Please see Padolski & Ye (DMTN-168) for more details on running with BPS and PanDA.

4 Sample BPS 'run' specification yaml

A 'run' is defined as a run of a BPS-run-spec.yaml through 'bps submit BPS-run-spec.yaml'.

This run-spec yaml file defines:

1. An existing (sub)pipeline definition to use for the run:

```
pipelineYaml: "$OBS_LSST_DIR/pipelines/imsim/DRP.yaml#step3"
```

These pipetasks may consist of several steps, and these steps may be defined in 'includes' in the DRP.yaml, for instance.

2. A target cluster to run on, and resource requests such as maximum memory (which can be different for different pipelines), number of cpus, and maximum wallclock time, and re-try attempt limits.
3. The location of the butler respository
4. an input collection to be extracted from the butler mentioned in the butler configuration along with a dataQuery which can restrict the inputs to a subset of the full input collection based on an SQL string (i.e. just do a few visits or a single tract or a single band)
5. various other configuration information including location of buckets, output path specs, etc

Here is an example look at a run specification yaml script. This one performs 'step3' pipeline tasks on a input collection of data. A full 'campaign' would be a set of perhaps 6 of these files, one for step1, one for step2, etc. Various critical name-value parameters are shown. Each run

in a campaign is (currently) driven by a single yaml-style configuration script which is launched by 'bps submit':

```
#includeConfigs:
#- ${CTRL_BPS_DIR}/python/lsst/ctrl/bps/wms/panda/conf_example/pipelines_check_idf.yaml

pipelineYaml: "$OBS_LSST_DIR/pipelines/imsim/DRP.yaml#step3"

payload:
  payloadName: pipelines_check
  runInit: true
#  output: "u/{operator}/{payload_name}"
  output: "u/{operator}/panda_test_ci_imsim_d_2021_09_15_w38"
  outCollection: "{output}/{timestamp}"
  butlerConfig: s3://butler-us-central1-panda-dev/dc2/butler.yaml
  inCollection: "2.2i/defaults/ci_imsim"
  dataQuery: "instrument='LSSTCam-imSim' and skymap='DC2' and (visit=256383 and detector=26)"
#  dataQuery: "tract = 9615 and patch=30 and detector IN (10..11) and instrument='HSC' and skymap='hsc_r

  sw_image: "lsstsqre/centos:7-stack-lsst_distrib-w_2021_38"
  fileDistributionEndPoint: "s3://butler-us-central1-panda-dev/hsc/{payload_folder}
/{uniqProcName}/"
  s3_endpoint_url: "https://storage.googleapis.com"
  payload_folder: payload
  runner_command: 'docker run --network host --privileged --env AWS_ACCESS_KEY_ID=$(
</credentials/AWS_ACCESS_KEY_ID) --env AWS_SECRET_ACCESS_KEY=$(</credentials/AWS_S
ECRET_ACCESS_KEY) --env PGPASSWORD=$(</credentials/PGPASSWORD) --env S3_ENDPOINT_UR
L=${S3_ENDPOINT_URL} {sw_image} /bin/bash -c "source /opt/lsst/software/stack/loadL
SST.bash;cd /tmp;ls -a;setup lsst_distrib;pwd;python3 \${CTRL_BPS_DIR}/python/lsst/
ctrl/bps/wms/panda/edgenode/cmd_line_decoder.py _cmd_line_ " >&2;'

#PANDA plugin specific settings:
ids_server: "https://aipanda015.cern.ch:443/ids"
placeholderParams: ['qgraphNodeId', 'qgraphId']

#IDF PanDA specific settings:
```

```
computing_cloud: LSST
```

```
#SLAC PanDA specific settings:
```

```
#computing_cloud: US
```

```
#computeSite: DOMA_LSST_SLAC_TEST
```

```
operator: huanlin    # defaults to login on submit machine
```

```
project: dev
```

```
campaign: quick
```

```
#....
```

5 Implementation Discussion Points

Here are some details of generating and tracking campaigns that will require further discussion regarding implementation details.

1. Ability to access up-to-date source of inputs to-be-processed into outputs and their current state (already processed, processed and flagged bad, in processing)
 - This could be done 'by hand' with butler queries initially to determine ranges of visits to process from specific date ranges (i.e. last night) or simulated data sets (i.e. DC2).
 - One will eventually need well defined, efficient ways to learn from the butler what is in there to be processed and what has already been processed, and if not, why not.
 - Efficiency is a consideration, as one does not wish to continually poll the butler or PanDA for the state of all past millions of visits, for example. This can be done with data to/from range terms in the PanDA query.
 - Helper scripts can assist with dumping dataids (or filenames with paths) and other info from the butler related to what is needed for campaign generation. This also helps when browsing logs and tracking down errors.
2. Ability to store and manage a set of BPS submit scripts (or equivalent).

- While one doesn't wish to reproduce the butler registry, there is a need to store somewhere campaign or run 'tuples' which can be a set of BPS submit file.yaml along with information such as time-of-submission and current status.
- The store of bps submit scripts won't have 'visit level' information, only 'ranges' as described in the SQL Butler query (visit-id between 236567 and 237673).
- For instance, as each BPS submit file.yaml is performed, the yaml script contents or a subset of 'tuples' from the yaml could be saved in a relatively lightweight database. Could sqlite or postgres be used for this? In Lim et al. (DMTN-181) it is proposed that there would be a new table in the butlerregistry schema that would hold 'DataID sets'. But perhaps an external lightweight database might be an alternative, as the butler will still be the ultimate source of metadata knowledge about input and output collections. The external database of tuples with BPS submit scripts don't need to have exact visit lists so long as the query used to generate the input lists are available.

Components of the tuple would be (input-query, bps-script, status,time-of-submission,...).

The bps-script file.yaml contains such things as the pipelineYaml (list of tasks to run), the sw-image (what software stack is being run). The site where the jobs are to be run (SLAC, IN2P3, UK, Cloud, user-laptop).

- The 'status' will need to be updated by getting information back from PanDA or Condor or whatever workflow system is being used whenever there is a change or at some interval.

3. Ability to watch, in realtime, as jobs or tasks within a campaign or set of campaigns are running.

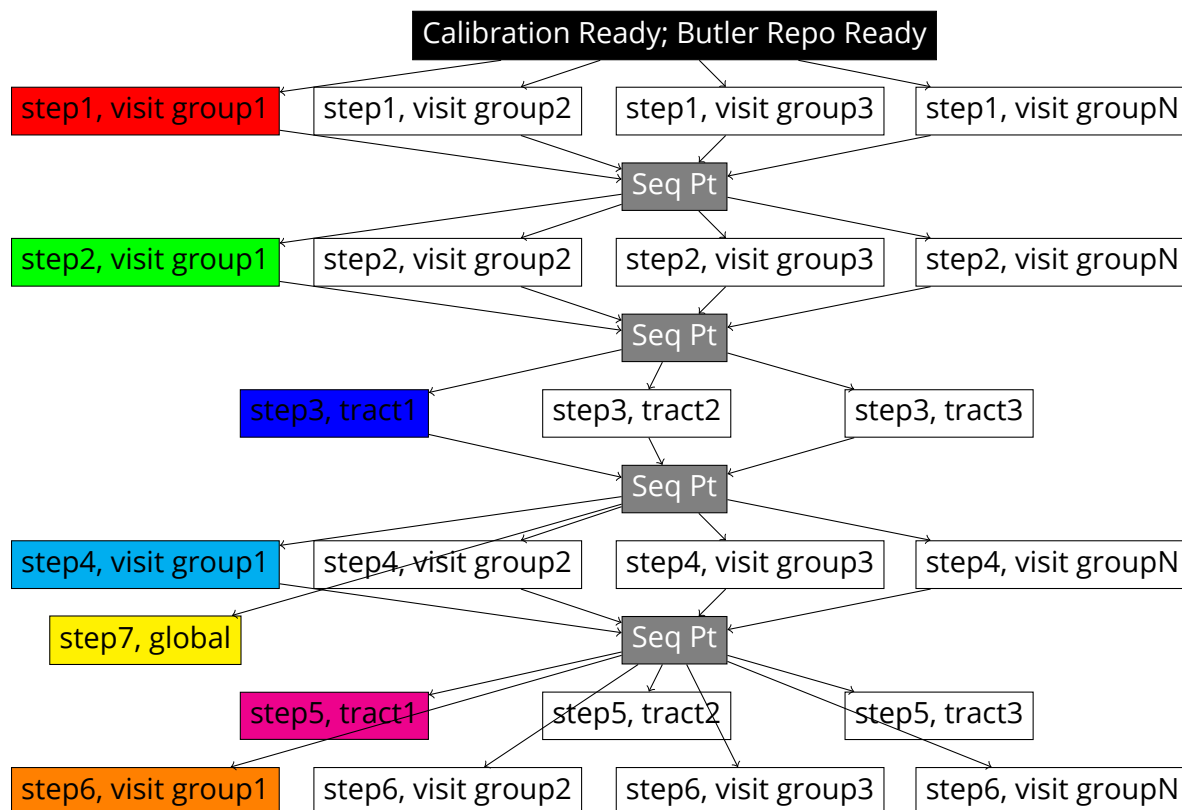
Here a web display that reads the lightweight database and shows/sorts the different running/recently-completed campaign components would be useful.

There is a tool called 'bps report' which does this at some level in the condor workload system, showing how many individual inputs in a set of jobs have been processed so far.

To work with PanDA there may need to be feedback from the PanDA system back to BPS or some other monitoring/tracking system, or 'deep links' into the PanDA IDDS could be generated for viewing.

6 Next steps in Campaign Tooling Development

A visual description of how a typical DRP campaign may be broken into related runs is shown in Fig. 6.



To begin, Campaign Tooling plans to develop tools to assist with campaign generation and tracking for the specific DP0.2 milestone of reprocessing the DC2 DESC data set in the iDF using BPS and PanDA O'Mullane & Dubois (RTN-013).

Figure 6 can serve as an initial guide to what is needed for generation of the DP0.2 campaign (left side of Figure 1): One run-spec.yaml file for each box in the Figure.

In order to implement the right side of Figure 1, one may need to setup a database to hold and visualize the run-spec definitions, as well as add interface items to BPS and PanDA to get feedback on running or submitted jobs. Some existing tools for generating and tracking workflows are Apache (airflow) and Mengel et al. (POMS) as well as the Pegasus Ensemble Manager Pegasus (PEM).

7 DRP.yaml steps

A recent version of the DRP.yaml file which define steps 1-7 in terms of pipelines/pipetasks units to be run:

```
description: DRP specialized for LSSTImSim
instrument: lsst.obs.lsst.LsstCamImSim
imports:
  # Inherits directly from pipe_tasks to avoid redefining sourceTable subset
  - location: $PIPE_TASKS_DIR/pipelines/DRP.yaml
tasks:
  isr:
    class: lsst.ip.isr.IsrTask
    config:
      connections.newBFKernel: bfk
      doDefect: False
      doBrighterFatter: True
  calibrate:
    class: lsst.pipe.tasks.calibrate.CalibrateTask
    config:
      connections.astromRefCat: 'cal_ref_cat_2_2'
      connections.photoRefCat: 'cal_ref_cat_2_2'
      astromRefObjLoader.ref_dataset_name: 'cal_ref_cat_2_2'
      photoRefObjLoader.ref_dataset_name: 'cal_ref_cat_2_2'
      python: >
      config.astromRefObjLoader.filterMap = {band: 'lsst_%s_smeared' % (band) for band in 'ugrizy'}
      config.photoRefObjLoader.filterMap = {band: 'lsst_%s_smeared' % (band) for band in 'ugrizy'}
  measure:
    class: lsst.pipe.tasks.multiBand.MeasureMergedCoaddSourcesTask
    config:
      connections.refCat: 'cal_ref_cat_2_2'
      match.refObjLoader.ref_dataset_name: 'cal_ref_cat_2_2'
      python: >
      config.match.refObjLoader.filterMap = {band: 'lsst_%s_smeared' % (band) for band in 'ugrizy'}
subsets:
```

step1:

subset:

- isr
- characterizeImage
- calibrate
- writeSourceTable
- transformSourceTable

description: >

Per-detector tasks that can be run together to start the DRP pipeline.

These may or may not be run with 'tract' or 'patch' as part of the data ID expression. This specific pipeline contains no tasks that require full visits. Running with 'tract' (and 'patch') constraints will select partial visits that overlap that region.

In data release processing, operators should stop to address unexpected failures before continuing on to step2.

step2:

subset:

- consolidateSourceTable
- consolidateVisitSummary
- makeCcdVisitTable
- makeVisitTable

description: >

Per-visit tasks that can be run together, but only after the 'step1'.

These may or may not be run with 'tract' or 'patch' as part of the data ID expression. Running with 'tract' (and 'patch') constraints will select partial visits that overlap that region.

This specific pipeline contains no tasks that require full visits.

This subset is considered a workaround for missing middleware and task functionality. It may be removed in the future.

step3:

subset:

- makeWarp

- assembleCoadd
- detection
- mergeDetections
- deblend
- measure
- mergeMeasurements
- forcedPhotCoadd
- transformObjectTable
- writeObjectTable
- consolidateObjectTable
- healSparsePropertyMaps
- selectGoodSeeingVisits
- templateGen

description: >

Tasks that can be run together, but only after the 'step1' and 'step2' subsets.

These should be run with explicit 'tract' constraints essentially all the time, because otherwise quanta will be created for jobs with only partial visit coverage.

It is expected that many forcedPhotCcd quanta will "normally" fail when running this subset, but this isn't a problem right now because there are no tasks downstream of it. If other tasks regularly fail or we add tasks downstream of forcedPhotCcd, these subsets or the tasks will need additional changes.

This subset is considered a workaround for missing middleware and task functionality. It may be removed in the future.

step4:

subset:

- forcedPhotCcd
- forcedPhotDiffim
- getTemplate
- imageDifference
- transformDiaSourceCat

description: >

Tasks that can be run together, but only after the 'step1', 'step2' and 'step3' subsets

These detector-level tasks should not be run with 'tract' or 'patch' as part of the data ID expression if all reference catalogs or diffIm templates that cover these detector-level quanta are desired.

step5:

subset:

- drpAssociation
- drpDiaCalculation
- forcedPhotCcdOnDiaObjects
- forcedPhotDiffOnDiaObjects

description: >

Tasks that can be run together, but only after the 'step1', 'step2', 'step3', and 'step4' subsets

This step includes tract-level aggregation Tasks. These should be run with explicit 'tract' constraints in the data query, otherwise quanta will be created for jobs with only partial visit coverage.

step6:

subset:

- consolidateDiaSourceTable

description: >

Tasks that can be run together, but only after the 'step1', 'step2', 'step3', and 'step4' subsets

This step includes visit-level aggregation tasks. Running without tract or patch in the data query is recommended, otherwise the outputs of consolidateDiaSourceTable will not contain complete visits.

This subset is separate from step4 to signal to operators to pause to assess unexpected image differencing failures before these aggregation steps. Otherwise, if run in the same quantum graph,

aggregated data products (e.g. diaObjects) would not be created if one or more of the expected inputs is missing.

A References

- [airflow]**, Apache, 2021, *Apache Airflow workflow generation and monitoring software*, airflow, URL <https://airflow.apache.org>
- [FARO]**, Bechtol, K., 2021, *Introduction to faro*, FARO, URL https://docs.google.com/presentation/d/102at3Vnz2k-9gxqy1mw-_2WzVGKMN7UP1R_XrXC7rpc/edit#slide=id.gdb10dd3f94_0_348
- [Metrics]**, Carlin, J., 2021, *Characteriation Metric Report: Science Pipelines Version 22.0.0*, Metrics, URL <https://dmtr-311.lsst.io>
- [DMTN-137]**, Chiang, H.F., Bektesevic, D., the AWS-PoC team, 2020, *AWS Proof of Concept Project Report*, DMTN-137, URL <http://DMTN-137.lsst.io>
- [DMTN-173]**, Economou, F., 2020, *The Observatory Logging Ecosystem*, DMTN-173, URL <https://dmtn-173.lsst.io>
- [iddefs]**, Jenness, T., 2020, *Gen 3 definitions of visit,exposure,observation id, etc*, iddefs, URL <https://community.lsst.org/t/change-to-visit-definition-for-lsstcam-and-latiss/4014>
- [LDM-633]**, Kowalik, M., Gower, M., Kooper, R., 2019, *Offline Batch Production Services Use Cases*, LDM-633, URL <https://ls.st/LDM-633>
- [LDM-636]**, Kowalik, M., Gower, M., Kooper, R., 2019, *Batch Production Service Requirements*, LDM-636, URL <https://ls.st/LDM-636>
- [DMTN-181]**, Lim, K.T., AlSayyad, Y., Bosch, J., et al., 2021, *Campaign Management*, DMTN-181, URL <https://dmtn-181.lsst.io/v/u-ktl-initial-draft/index.html>
- [POMS]**, Mengel, White, Podstavkov, et al., 2020, *Production Operations Management System (POMS) for Fermilab Experiments*, POMS, URL https://www.epj-conferences.org/articles/epjconf/pdf/2020/21/epjconf_chep2020_03024.pdf

[RTN-001], O'Mullane, W., 2020, *Data Preview 0: Definition and planning.*, RTN-001, URL <http://RTN-001.lsst.io>

[RTN-013], O'Mullane, W., Dubois, R., 2020, *Near term workflow for pre-operations with PanDA*, RTN-013, URL <http://RTN-013.lsst.io>

[DMTN-168], Padolski, S., Ye, S., 2021, *Running Science Pipelines using PanDA*, DMTN-168, URL <https://dmtn-168.lsst.io>,
LSST Data Management Technical Note

[PanDA], PanDA, 2021, *Production and Distributed Analysis*, PanDA, URL <https://panda-wms.readthedocs.io/en/latest>

[PEM], Pegasus, 2013, *Pegasus Ensemble Manager*, PEM, URL <https://confluence.pegasus.isi.edu/display/pegasus/Ensemble+Manager>

B Acronyms

Acronym	Description
BPS	Batch Production Service
DC2	Data Challenge 2 (DESC)
DESC	Dark Energy Science Collaboration
DF	Data Facility
DM	Data Management
DMTN	DM Technical Note
DMTR	DM Test Report
DP0	Data Preview 0
DR1	Data Release 1
DRP	Data Release Production
FGCM	Forward Global Calibration Model
FTS	File Transfer Service
HSC	Hyper Suprime-Cam
IDF	Interim Data Facility
IN2P3	Institut National de Physique Nucléaire et de Physique des Particules
IP	Internet Protocol
LDM	LSST Data Management (Document Handle)

LSST	Legacy Survey of Space and Time (formerly Large Synoptic Survey Telescope)
PanDA	Production ANd Distributed Analysis system
RTN	Rubin Technical Note
SLAC	SLAC National Accelerator Laboratory
SQL	Structured Query Language
SST	Subsystem Science Team
UK	United Kingdom
URL	Universal Resource Locator
US	United States
bps	bit(s) per second
stdout	standard output