May 2022 Performance Sprint Summary

Executive Summary

Alert Production has a high-level requirement to have transmitted alerts to community brokers 60 seconds after image readout is complete. In order to avoid premature optimization, to date the AP team has focused on algorithmic performance and functional completeness. However, as we near the end of construction, increased attention to processing latency is necessary.

In May 2022, we conducted a two-week sprint to assess current processing latencies. Profiling revealed several configuration changes which provided substantial speedups. We also developed improved datasets and analysis tooling to support future optimization. Current pipeline runtime on DC2 simulated LSST data takes about 300 seconds per detector on `main`, and about 200 seconds when we revert to the psfex psf determiner from the current PIFF default (see RFC-857).

Ongoing work will be needed to ensure we reach the latency requirements in production.

Sprint Goals and Organization

During May 2022, we undertook a sprint to assess current pipeline execution speed. We chose to use the DC2 simulated LSST images as they have the same size in pixels and object density as real LSST images. Our goals were to conduct both detailed profiling on runs of a few detector-visits as well as bulk timing of a larger sample. We also aimed to facilititate monitoring realistic latencies in an automated manner (through CI) and to improve the realism of our estimates in the absence of 12-month lightcurve history.

Sprint participants were Meredith Rawls and John Parejko , with Eric Bellm supervising the overall effort.

Progress was tracked on DM-34623 - Jira project doesn't exist or you don't have permission to view it. .

Accomplishments

- Removed `CharacterizeImage` plugins that are unnecessary for AP, resulting in a ~30% improvement in runtime.
- Identified `PiffPsfDeterminer` as a substantial contributor to `CharacterizeImageTask` time; plan to switch back to psfex, which should be sufficient to our needs.
- Performed first timing tests of ap_association at relevant scales (100+visits)
- Pair-coding discussion identified further inefficiencies in `CharacterizeImageTask`, now RFC'd for improvement

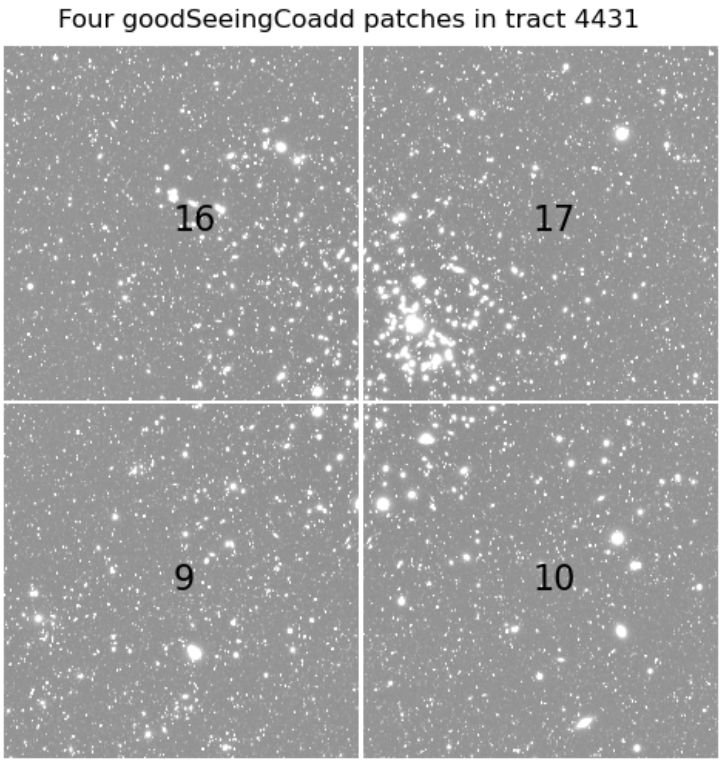
  RFC-857 - Jira project doesn't exist or you don't have permission to view it.
- Started to build a tool for looking at the output of ap_verify metrics outside of Chronograph, and to compare different ap_verify runs. See

  DM-34865 - Jira project doesn't exist or you don't have permission to view it.
- Made a framework for a DC2 ap_verify ci test dataset, but got stuck on exporting appropriate calibs. See

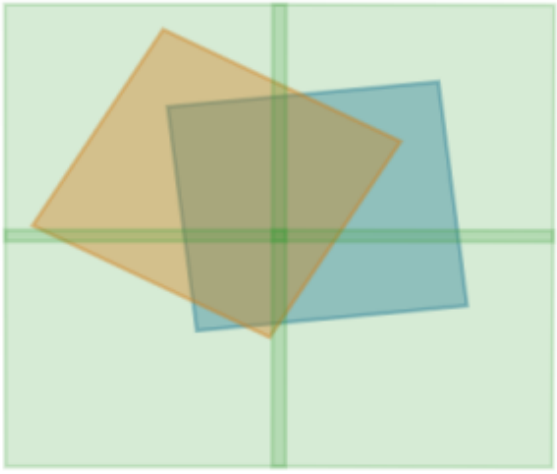
  DM-34845 - Jira project doesn't exist or you don't have permission to view it.
- Explored pyinstrument for statistical profiling, but did not see much advantage over the standard python profiler.
- Expanded python profiling section of the developer guide.

Dataset

We selected four patches in DC2 patch 4431, which was processed by Kenneth Herner in the shared `/repo/dc2` collection `u/kherner/2.2i/runs/tract4431-w40`. This tract has 10-year depth, which guarantees there will be significant visit overlap in any given patch. We chose patches 9, 10, 16, and 17, which are adjacent to one another, and includes a slightly dense region that appears to be a cluster of galaxies.



We next wrote a selector script (**final landing place TBD**) to identify all visit+detector datasets that fall **entirely** within these four patches. This was important because DC2 is highly dithered, so there is no way to choose a list of visits and assume significant spatial overlap. As an example, the two exposures shown below (blue and orange) overlap each other and fall within the four patches (green), and we intend to use them in a new ap_verify-style CI dataset. The blue dataset is visit 982985, detector 164 and the orange dataset is visit 943296, detector 168.



The datasets we curated are in the shared `/repo/dc2` as TAGGED collections named `u/mrawls/DM-34827/defaults/4patch_4431` (for the raws and all necessary supplementary inputs, i.e., calibs, refcats, and skymaps). The coadds we used as templates for running the AP Pipeline are of type `goodSeeingCoadd` and originate from the `u/kherner/2.2i/runs/tract4431-w40` collection. Just the datasets needed were curated into the `u/mrawls/DM-34827/coadd/4patch_4431` collection. We created this — and intentionally did NOT chain it along into `defaults` — because in the future, we may want to build different templates, and to avoid unintentionally chaining along oodles of datasets we did not want to process.

Processing

We used `bps` on `lsst-devl01` to process the larger 272-visit set of images through the ApPipe pipeline. We repeated the process using both w.2022.21 and w.2022.22. The latter included new changes to `CharacterizeImage` as described below. The pipeline was the standard ApPipe with an APDB created in advanced and configured at runtime, plus it also had a second yaml import from `ap_verify` designed to compute runtime metrics for each task.

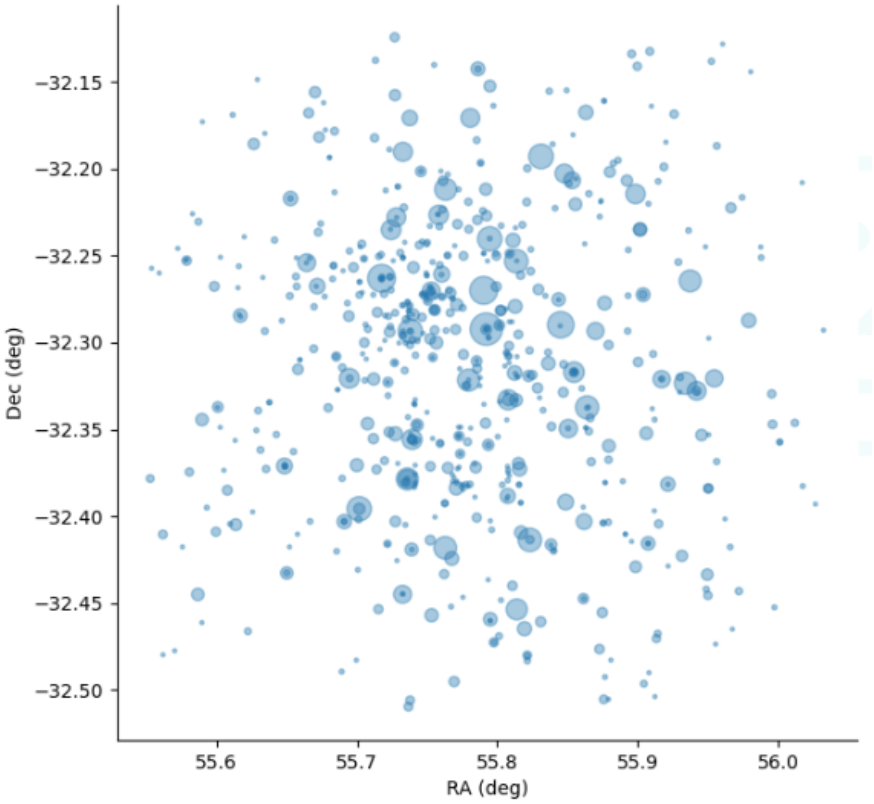
```
description: AP pipeline for LsstCam-imSim, with ap_verify runtime metrics
# (1) Execute `make_apdb.py`
# (2) Run this pipeline, setting appropriate diaPipe configs
#     (diaPipe configs should match the make_apdb.py configs)

instrument: lsst.obs.lsst.LsstCamImSim

imports:
  - location: $AP_PIPE_DIR/pipelines/LsstCamImSim/ApPipe.yaml
  - location: $AP_VERIFY_DIR/pipelines/MetricsRuntime.yaml

tasks:
  diaPipe:
    class: lsst.ap.association.DiaPipelineTask
    config:
      apdb.db_url: 'postgresql://mrawls@lsst-pg-devel1.ncsa.illinois.edu/lsstdevapdb1'
```

The `bps` submit yaml used was, e.g.,

```
pipelineYaml: '/project/mrawls/ap_profile/ApPipe_DM-34828.yaml'
project: ApPipe_4patch_DC2
campaign: DM-34924

computeSite: ncsapool
requestMemory: 4096
includeConfigs:
  - ${CTRL_BPS_DIR}/python/lsst/ctrl/bps/etc/bps_defaults.yaml

payload:
  payloadName: DM-34828-w_2022_22
  #payloadName: DM-34828-w_2022_21
  butlerConfig: /repo/dc2/butler.yaml
  inCollection: u/mrawls/DM-34827/defaults/4patch_4431,u/mrawls/DM-34827/coadd/4patch_4431
  dataQuery: instrument = 'LSSTCam-imSim' and skymap='DC2'

pipetask:
  imageDifference:
    requestMemory: 8192
```

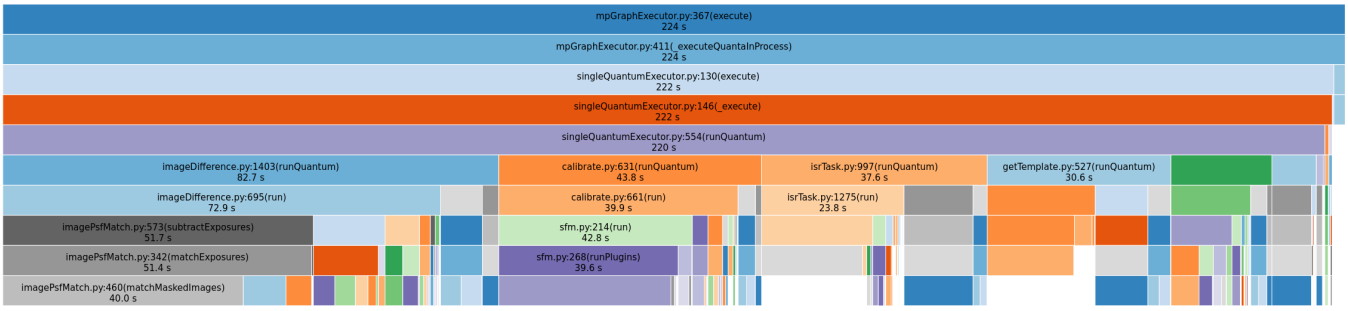
We encountered a middleware bug, which was subsequently resolved (DM-34924). The processing completed within about an hour. The output collections are `u/mrawls/DM-34828-w_2022_21` with APDB postgres schema `mrawls_DM34828_w202221` and `u/mrawls/DM-34828-w_2022_22` with postgres schema `mrawls_DM34828_w202222` . The only failed dataset was a single catastrophic astrometry failure in the latter, which we have yet to investigate.

We verified the dataset processed in a reasonable fashion by plotting DIA Objects on the sky. Larger circles correspond to DIA Objects composed of more DIA Sources. The source density is higher near the middle of the spatial region, because our visit+detector selection criteria only chose datasets that **fully** overlap the four template patches, and therefore the center has the greatest overlap.

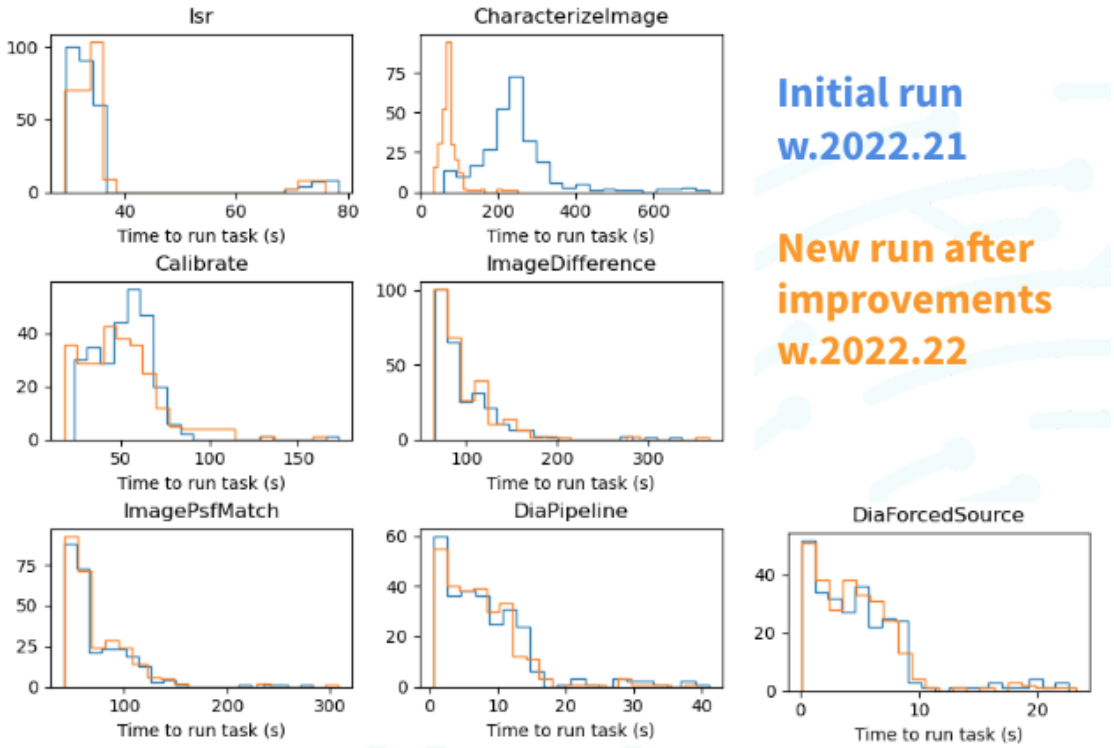


Results

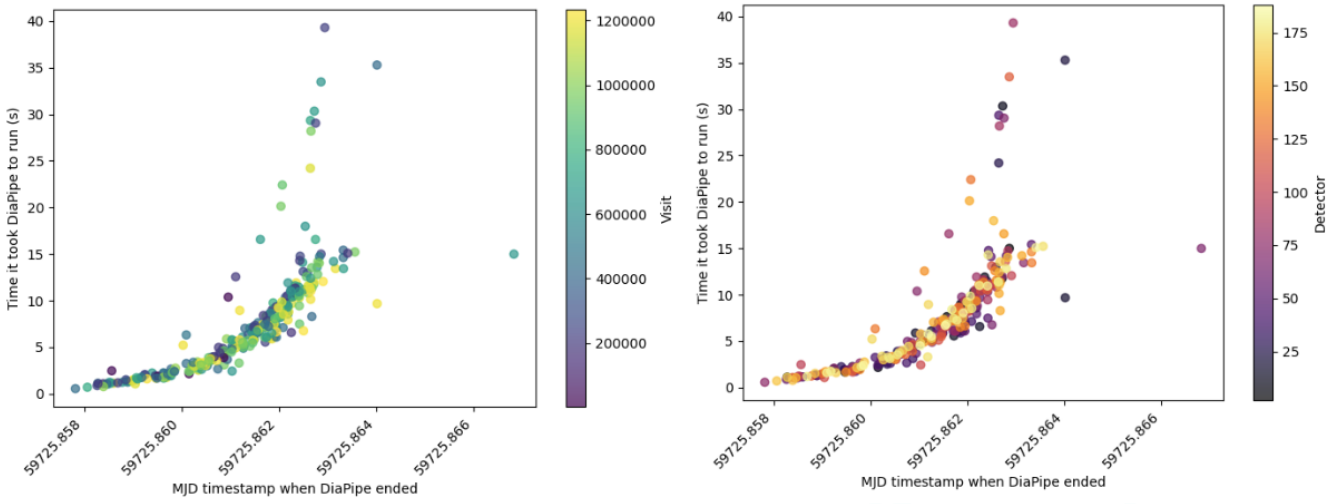
See single detector call graph below for the longest runtimes after the above improvements were made. The green box on the right is where CharacterizeImage is now, after plugin removal and using psfex in `MeasurePsfTask`. Reading data contributes ~30 seconds to this execution time, which should not be an issue in the prompt processing environment, as we will be prefetching most data, and using an in-memory butler.



The figure below shows runtime comparisons for each ApPipe task for the two 272-visit bps runs. The biggest difference is in `CharacterizeImageTask` .



The figure below explores how DiaPipe, the task which does DIA Source association into DIA Objects and performs DIA forced photometry, is slower for datasets processed later. The two panels are identical, except the left is colored by visit number and the right is colored by detector number. Both are for the second large run, but we did make plots for both, there was no appreciable difference between the w.2022.21 and w.2022.22 runs.



Finally, to explore further the subtasks done by DiaPipe, we investigated more granular timing metrics. The plot below reveals that forced photometry dominates the runtime, with catalog loading coming in second. While it is reasonable for both of these to become slower over time as the DIA Source history grows larger, further work needs to be done to see if we can gain any improvements in forced photometry.

Caveats

- These tests were performed with the old `ImageDifferenceTask`; we expect the new `SubtractTask` to be at least somewhat faster due to fewer warping steps.
- We performed these tests on NCSA hardware, which does not match that of the USDF production environment (computing hardware or databases).
- Execution in operations will include a pre-loading step for calibrations and lightcurve history, before the 60-second clock begins.
- Our datasets did not include the 12-month lightcurve history that will be present in steady-state operations.
- The existing timeseries features computed on the lightcurves are not final, and are generally less CPU-intensive than those baselined.

Future Work

- Why is ISR runtime bimodal? What causes the variance in ISR runtime for faster runs?
- Does psfex produce good enough PSFs for AP, given it's much faster runtime compared with piff?
- What other measurement plugins can we remove (see RFC-857 for a minimal end-goal)?
    - How do we measure the runtime of C++ sfm plugins? They don't obviously show up in the call stacks in snakeviz.
    - Test not iterating the PSF fitter in characterizeImage (psfIterations=1), and also disable all the plugins during that psf characterization (this is part of RFC-857).
- Why did calibrate take somewhat longer with the old characterize plugins in place?
- Run line_profiler on each of our Tasks (isr, characterize, calibrate, diffim, getTemplate) to better pin down specific slow steps.
- Run profiling tests on the new diffim, and compare it with the current one (I expect the new one is faster because it should do one less warping step).
- Finish making the ap_verify_ci_dc2 dataset to simplify testing.

  DM-34845 - Jira project doesn't exist or you don't have permission to view it.
- Investigate scaling issues in DIAForcedSource computation

  DM-35158 - Jira project doesn't exist or you don't have permission to view it.