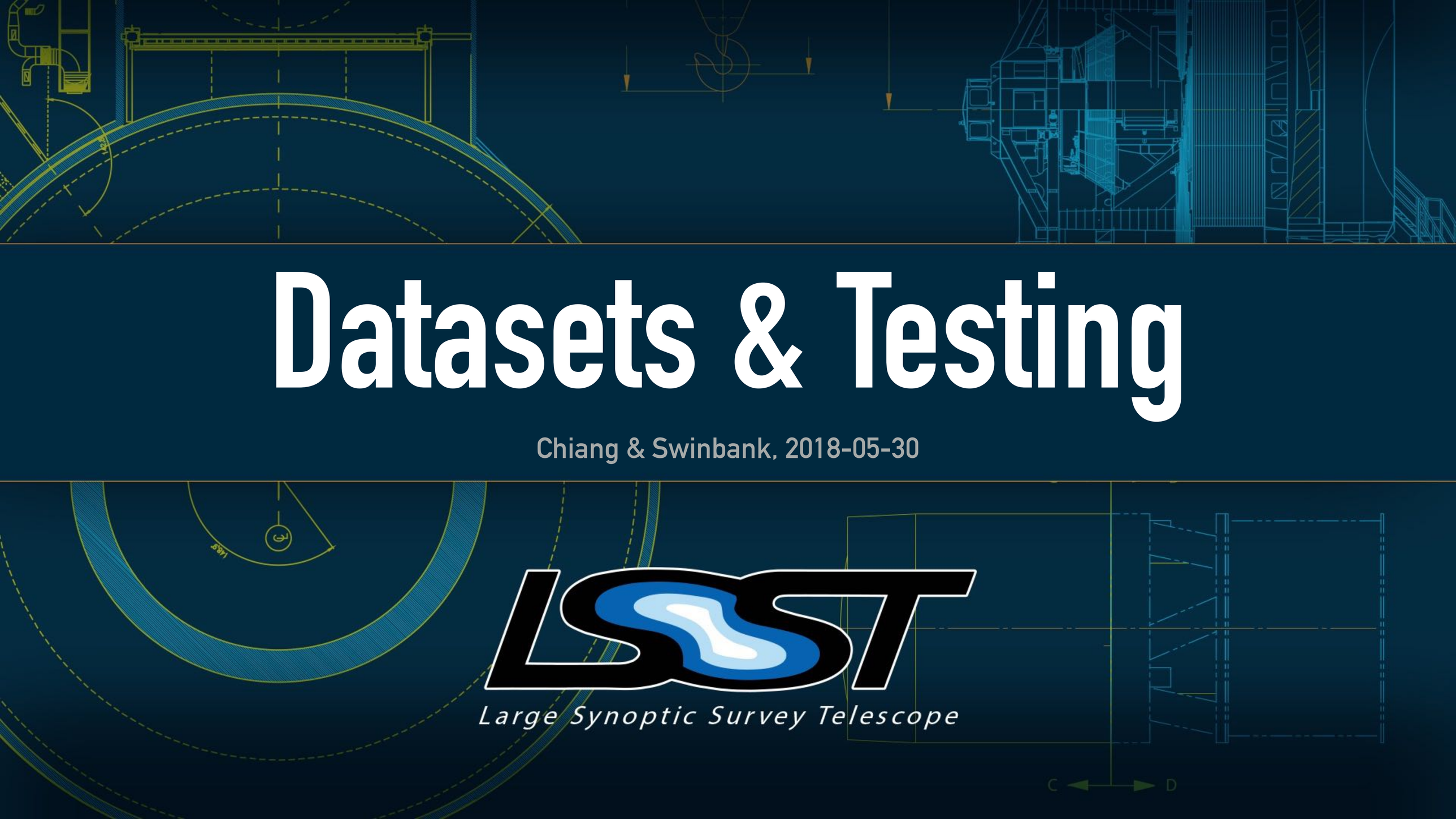
Datasets & Testing

Chiang & Swinbank, 2018-05-30



Notes

- Collected a bunch of rough notes & comments at https://confluence.lsstcorp.org/pages/viewpage.action?pageId=73581056
- Started writing these up into a more coherent document.
  - https://github.com/lsst/qawg-test-report; ping me if you want access.
  - I found this exercise useful, in that it forced me to think through issues.
  - It also took *a lot* of time, and is nowhere near done.

Unit Tests

- We do not propose any significant changes to the way we handle tests.
  - We're a little uncomfortable that our tests confuse unit & integration testing, but we don't think it's plausible to imagine we'll address that.
- We suggest a number of improvements to the way we handle CI.
  - Things like tracking code coverage, providing clearer guidance for developers about what they're supposed to do.
  - None of this is new or innovative; pretty sure SQuaRE already have much of it on their radar, but we think/hope that making them explicit recommendations of the WG will help.
  - Concerned that we need to make sure examples and executables are tested.
    - That's a prioritisation issue for SQuaRE development.
    - Also proposing an aggressive approach to eliminating broken examples.

Integration Tests

- Spent some time trying to understand what's currently happening with respect to integration testing... there are a lot of overlapping-but-different approaches.
  - validate_drp, ci_hsc, lsst_dm_stack_demo, lsst_ci, lsst_qa, ci_ctio0m9, ap_verify...
- Would like to define a clear hierarchy of what sort of tests are run when
  - e.g., tiny integration test on every merge, nightly larger scale testing, bi-weekly medium scale test, periodic large scale “data challenge” like HSC PDR1.
  - But we've not got as far as developing a coherent picture yet.
- Would like to have all these tests *look the same*, i.e. run in a similar framework rather than the current mixture of shell scripts, SConstruct files, etc.
  - This enables code reuse, lowers the threshold for developers, etc
  - However, not clear whether we want to do this *now*, or if we should live with the current chaos until some future SuperTask-based promised land.
- Wondering about the workflow system / execution engine for these tests.
  - Should some tests run on Jenkins workers, and others on the verification cluster? Or do we do everything on the VC?
  - When using the VC, does Jenkins still start jobs, collect results, etc?
  - Which jobs require manual intervention?

Datasets

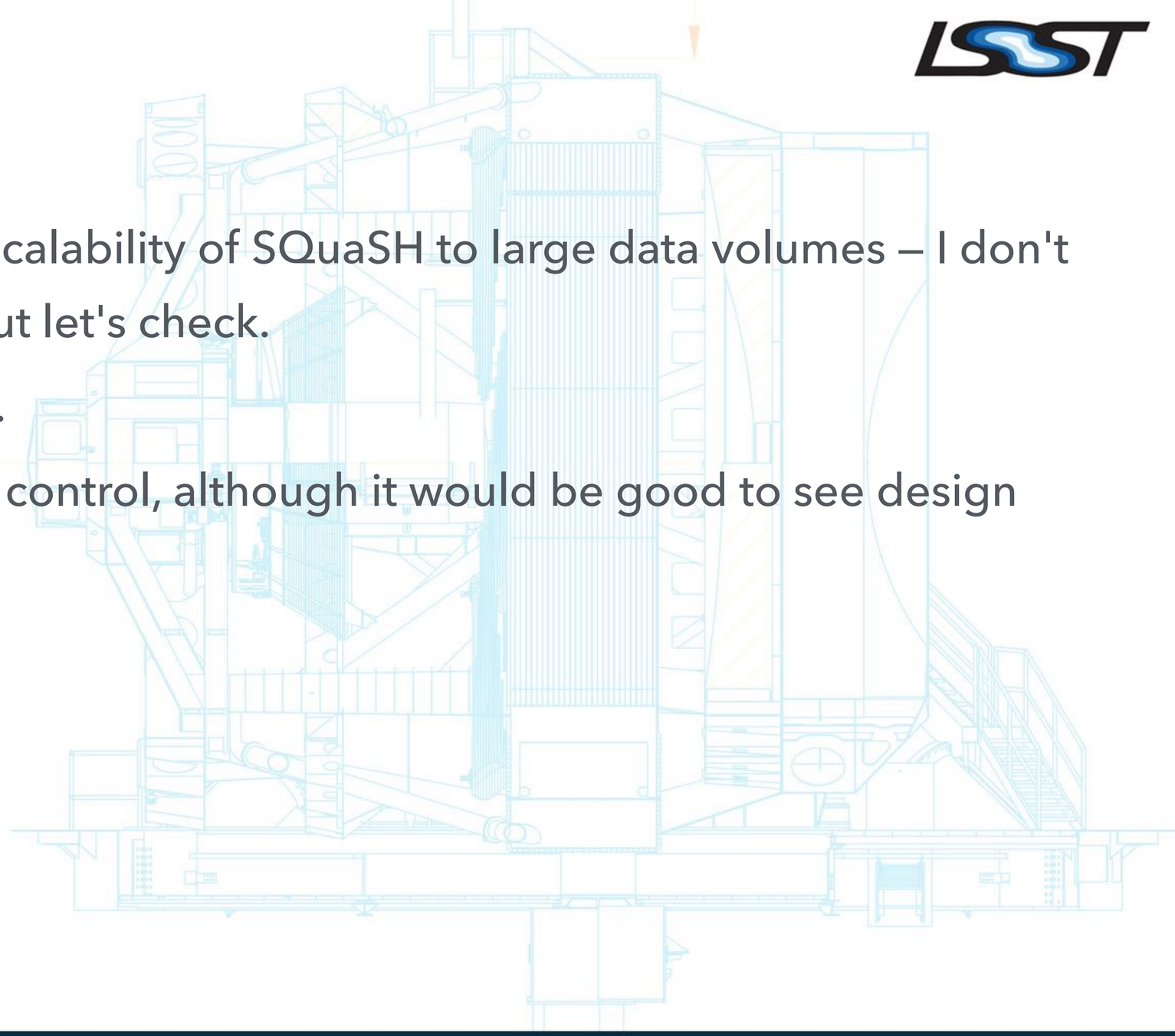
- After some discussion, we think it is valuable to make large datasets available on the VC/GPFS and smaller datasets on Git LFS (as we do now).
- We want to propose a common structure & format for all the Git LFS datasets.
  - Relates to common structure for integration tests.
  - We note the AP team gave a lot of thought to how dataset packages interact with ap_verify. Want to see if we can reuse that model elsewhere, but haven't yet got to grips with it.
- Many of our LFS datasets contain both raw & processed data.
  - We wonder how useful the processed data is – in particular, at the moment it's usually outdated and unusable with recent stack releases, and the sky hasn't fallen.
  - If it is useful, we think it's essential to introduce an automated process for keeping it up to date (e.g. reprocessing & committing it as part of the weekly stack release).

Datasets (2)

- We propose that all stored datasets (especially on GPFS) have named product owners.
  - Product owners are responsible for ensuring that datasets are well described, compatible with recent stack versions, etc.

Metric Tracking

- Some question raised about the scalability of SQuaSH to large data volumes — I don't think this should be a problem, but let's check.
- Otherwise, excited by DM-14328.
- Seems like Angelo has this under control, although it would be good to see design documentation.



Run Time Performance

- There's already a bunch of monitoring tools at the Data Facility.
  - https://confluence.lsstcorp.org/display/~sthrush/Node+Utilization+for++HSC-RC2+Reprocessing+Jobs is super neat.
- Ultimate goal, though, is only partially addressed by these: want to provide immediate (well, CI-timescale) feedback to developers on the impact of algorithmic changes.
- Simply tracking timings from ci_hsc (or descendent thereof) in a SQuaSH like interface may be adequate.

More Datasets!

- We have input from the SST about datasets they want to use for large scale testing.
- We should try to generate a set of generally useful datasets distributed by Git LFS for smaller scale tests.
  - Covering as many developer use cases as are obvious
  - But given the standardization we discussed earlier, the overheads for adding new datasets as we go should be low.