Interfacing to Real Data

TOAST Workshop - 2019 Trieste

Raw Data

- Usually a binary dump
  - CData
  - GCP / Archive files
  - Telemetry packets
  - spt3g (small frames)
- Naturally frame-like, since that is how most data acquisition systems work
- Often multiple sources of information from different systems

Data for Analysis

- Usually some merging of raw data sources
- Usually "more unpacked" (longer contiguous chunks)
  - Dirfiles (subdirectories, zipped archives)
  - Spt3g (with bigger frames)
  - HDF5 (2D datasets)
  - FITS (binary tables)
- Usually has been "indexed" somehow. Metadata like time range, sky patch, weather properties, etc are stored in a DB or some other format.

Data for Analysis

- Frequently data has natural split in time and detector directions
  - (Time) Changes in pointing configuration
    - Elevation changes
    - Thruster firing
  - (Time) Cooler cycling / thermal changes
  - (Time) Computer reboots
  - (Detector) Different readout cards
  - (Detector) Split by frequency
- These natural splits should be used to define TOAST observations.

TOAST Considerations

- Determine observation boundaries
- What (python) tools does your project have for querying metadata? Can you select data based on the observation splits you want?
- In order to read telescope and detector data for an observation, what information is needed? Examples:
    - Paths to files
    - Data ranges within files
    - Other metadata that is not contained in the files

TOAST Interface

- TOD class which takes in the constructor all information to read data for the observation.
- Python code to query metadata sources and instantiate observations:

```
def create_observations(selection, commworld):
    if commworld.rank == 0:
         # Only one process touches DB
        allobs = query(selection)
     allobs = commworld.bcast(allobs, root=0)
     comm = toast.Comm(world=commworld)
     # Distribute observations based on some criteria (i.e size)
     groupobs = distribute_obs(comm, allobs)
     for obs in groupobs:
        obs["tod"] = MyTOD(....)
```

TOAST Interface Example: Simons Observatory

Most S.O. code is publicly developed in the model of LSST, DESI, etc. Currently only real hardware information and some small scripts are private. Enables integration with services (e.g. CI, readthedocs).

Example: Loading observations of realistic (simulated) data:

https://github.com/simonsobs/sotodlib/blob/master/sotodlib/data/toast_load.py

WARNING: all this is code under active development. It is very specific to the proposed S.O. data format / schema. It is just a concrete example.