1.08.02
Observatory Control System

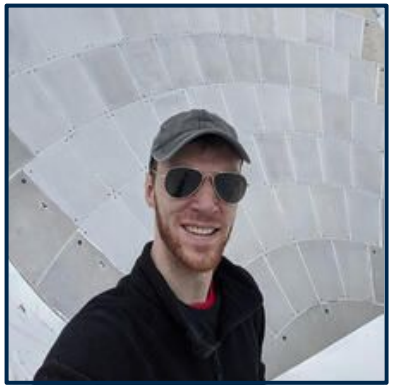
Brian Koopman

CMB-S4 DAQ Conceptual Design Review
July 24, 2023

Who Am I

Brian Koopman

1.08.02 - Observatory Control System Lead



- Research Scientist @ Yale
- Design and development of DAQ, monitoring, and control software systems for the Simons Observatory. Core developer of the Observatory Control System. (3.5 years)
- PhD working on ACT with a focus on polarization calibration, detector testing, and improving the remote observing experience through development of web based tools. (6 years)

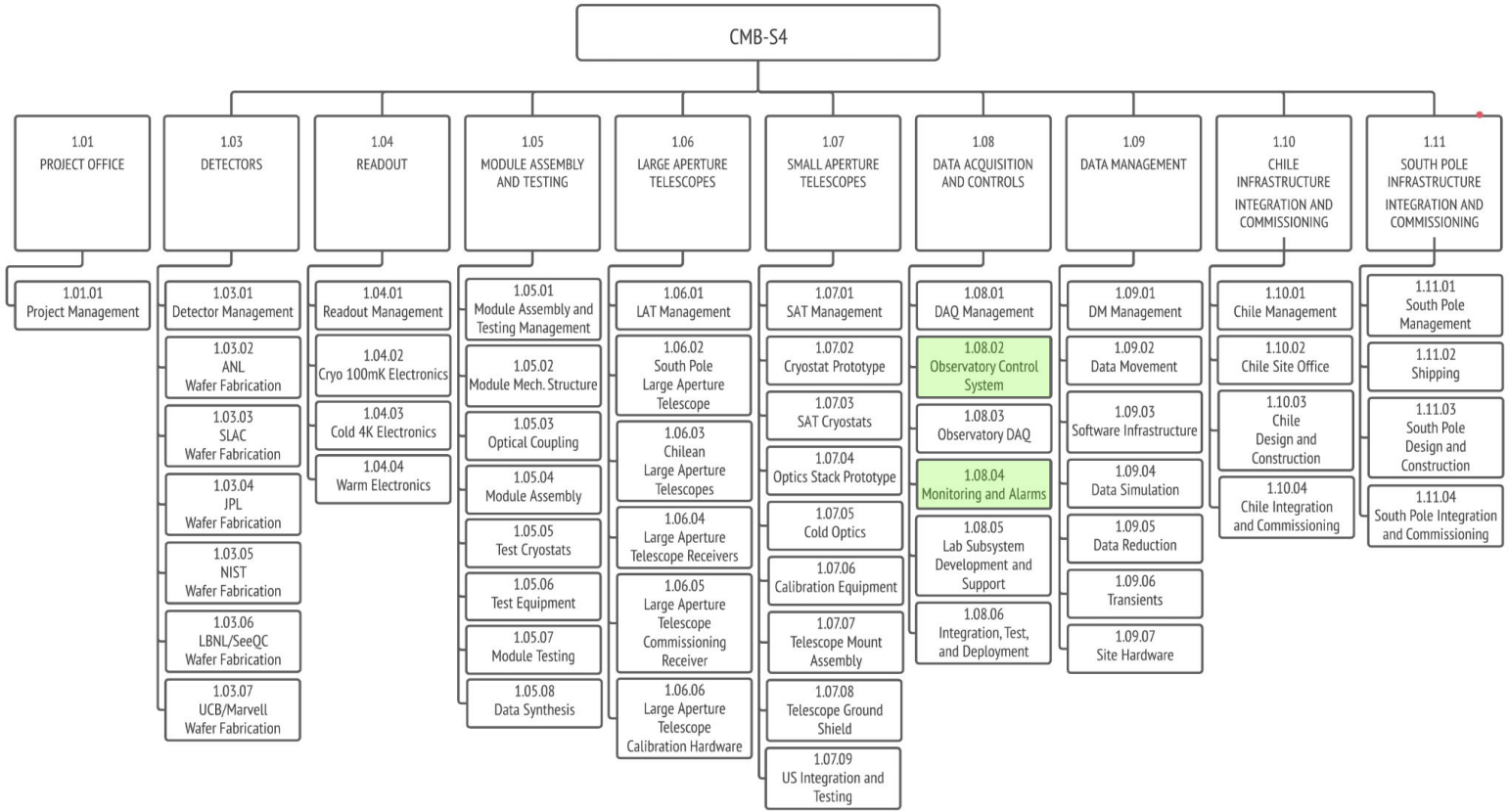
Outline

- Updates since last review
- Locations in WBS
- Scope
- Design drivers and interfaces
- OCS Overview
  - Architecture
  - OCS Agents
  - OCS Clients
  - Network Configuration
  - Access Control
  - Agent Unit and Integration Testing
  - Documentation
  - Example Configuration Layouts
- Next steps towards CD-1
- Summary

Updates Since Last Review

- Changes to OCS
  - HostManager Agent can now control agents running in Docker
  - OCS plugin system created – enables easy integration/organization of S4 ocs agents
  - Task abort now supported
  - Updated Agent development guide for new users to follow for writing an agent
  - InfluxDB publisher performance improvements
  - Many more agents written
- Scheduled observation infrastructure
  - Nextline – the SO “sequencer” exists
    - Runs schedules that dictate daily observations
    - Web interface for users to load and run schedules w/history viewer of past schedules
  - sorunlib – high level API for schedules to use to control observatory (collection of OCSClients)
  - Scheduler
    - Code which translates an observation strategy into sorunlib based schedules which nextline can run
    - Web server for nextline to fetch schedules from automatically
  - All comes together to allow fully remote and autonomous observations
- SO is also now in the process of being deployed, including official OCS deployment in the field

Location In The CMB-S4 Work Breakdown Structure



Scope

- Design and develop control software for commanding and monitoring hardware in the lab and on site.
- Scope includes interfacing the control system with all hardware, including the bolometers, telescope platforms, cryogenics, and housekeeping equipment.
- Table of L4s:

1.08.02	Observatory Control System	L3
1.08.02.01	Bolometer Readout Control	L4
1.08.02.02	Telescope Platform Control - SP, Chile, LAT, SAT	L4
1.08.02.03	Cryogenic Control	L4
1.08.02.04	Housekeeping Control	L4
1.08.02.05	Observatory Subsystem Control	L4
1.08.02.06	Observation Scheduling	L4
1.08.02.07	Control Framework	L4

Design Drivers

- Provide distributed control and monitoring of hardware, with a messaging and routing layer designed for distributed systems.
- Provide local and remote user interaction with all hardware on the observatory control system network.
- Control software must be flexible enough to scale from small in lab deployments to full site scale deployments.
  - Provides thorough testing of the software before site deployment
  - Familiarizes users in labs with the software
- Configuration of the software should be easily managed by the user.
- The control system framework should be easy to add new components to, allowing qualified users developing hardware to easily integrate with the software.
- The software should be written in a small set of well-known languages, namely C++ and Python, allowing broad contributions from users in labs.
- The software should run on a wide variety of hardware and operating systems, allowing some flexibility in labs, where computing hardware will be varied.

Interfaces With Other L3s

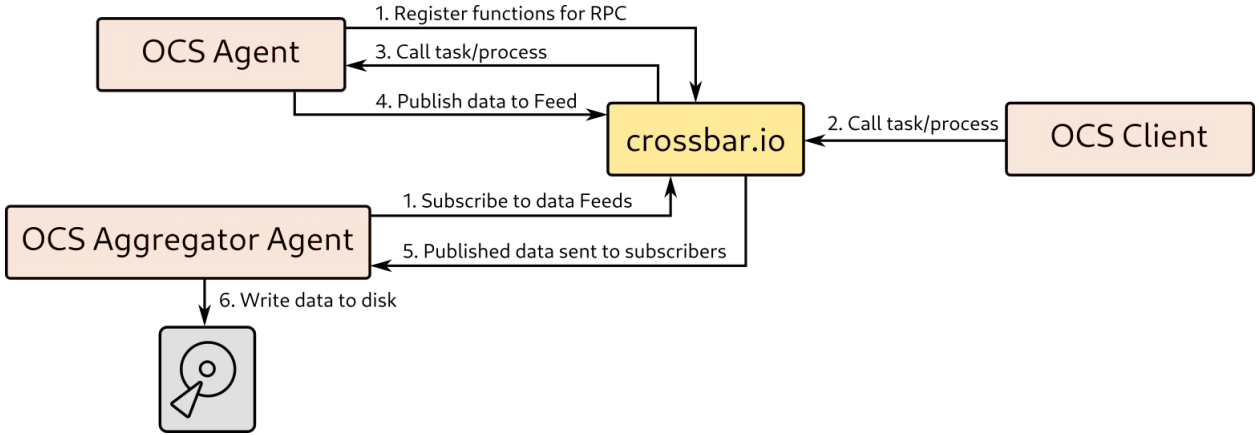
- Interface with 1.08.03 (DAQ):
  - Data passed to DAQ for aggregation to disk
- Interface with 1.08.04 (Monitoring):
  - Low-speed housekeeping data streams made available to monitoring and alarm systems
- Interface with 1.08.05 (Lab Development and Support):
  - OCS software provided for lab use, including documentation for user support
- Interface with 1.08.06 (Integration, Test, and Deployment):
  - OCS software delivered for integration, testing, and deployment

OCS Overview

- The Observatory Control System (OCS) is a distributed control system designed to coordinate DAQ across an observatory, originally designed and built for use on Simons Observatory.
- Design goals:
  - Easy to use
  - Easy to add additional components to for new hardware
  - Scalable from small lab deployments to full site deployment
- Mostly written in Python, with use of some open source systems
- Main messaging and data passage system is built on the open source/commercial crossbar.io platform
- Use additional open source platforms for live and historical data viewing
  - Grafana + Loki - Web based plotting and observability software, and log aggregation
  - InfluxDB - Time series database backend for Grafana

OCS Architecture

- Two main components to OCS:
  - Agents - long running software servers that interface with hardware or other software
  - Clients - scripts that orchestrate the actions of one or more OCS Agents
- Agents make available two types of commands:
  - Tasks - function that ends in finite time
  - Processes - function that runs for an open-ended amount of time, until stopped by a client



OCS Agents

- Small set of “core” OCS Agents - ocs repo
  - HK Aggregator Agent - writes all data from all HK Agents to disk in .g3 format
  - InfluxDB Publisher Agent - writes all data (possibly downsampled) from HK Agents to InfluxDB
  - Registry Agent - Keeps track of all other running OCS Agents on the network and the status of their tasks/processes
  - Host Master Agent - Can control startup and shutdown of Agents depending on OCS configuration
  - Fake Data Agent - Simulated data Agent used for testing
- Large set of Simons Observatory Control System (socs) Agents - socs repo
  - Provides functionality specific to SO hardware
  - Over half have been user contributed (demonstrating ease of use among labs)
  - Major Agents are functional and are being tested
    - ACU - telescope pointing
    - CHWP - control and readout of the CHWPs
    - SMuRF - control and monitoring of the detector readout crates and collecting output files

SOCS Agents

Table 1. Summary of current and in development `ocs` Agents. Core `ocs` Agents are marked by an asterisk. All other Agents listed are kept with `socs`.

Agent	Hardware/Software	Development Status	Description
AggregatorAgent*	Housekeeping Archive	In use	Saves HK data from feeds to `.g3` files in the HK archive.
BluteforsAgent	Bluefors LD400 Dilution Refrigerator	In use	Parses and passes logs from LD400 software to `ocs`.
CryomechCPAAgent	Cryomech CPA Model Compressors	In use	Communicates with compressor over Ethernet to record operations statistics. On/Off control still in development.
CHWPAgent	Custom Built Cryogenic HWP	Under development	Command and control cryogenic half wave plate (CHWP) hardware.
DSEAgent	Deep Sea Electronics Controller	Under development	Collect diesel generator statistics over modbus interface.
HostMasterAgent*	`ocs` Agents	In use	Optional Agent to start and stop Agent instances. Useful for running Agents outside of Docker containers.
iBootbarAgent	iBootBar PDUs	In development	Monitor and control managed power distribution units (PDUs) for remote power cycling of electronics.
InfluxDBAgent*	Influx Database	In use	Record HK data feeds to Influx database for viewing in Grafana.
Lakeshore372Agent	Lakeshore 372 Resistance Bridge	In use	100 mK thermometer readout and heater control for dilution refrigerator operation.
Lakeshore240Agent	Lakeshore 240 Input Module	In use	1K-300K thermometer readout.
LabJackAgent	LabJack T4/T7	In use	Communicates with a LabJack over Ethernet for generic DAC. Used in warm thermometry readout.
PfiefferAgent	Pfieffer TPG 366	In use	Six channel pressure gauge controller readout over Ethernet. Used to monitor pressures within vacuum systems of the dilution refrigerators.
MeinbergM1000Agent	Meinberg LANTIME M1000	In use	Monitor health of Meinberg M1000 timing system via the Simple Network Management Protocol (SNMP).
RegistryAgent*	`ocs` Agents	Under development	Tracks the state of other running `ocs` Agents on the network.
SCPIPSUAgent	SCPI Compatible PSUs	In use	Command and control power supplies compatible with Standard Commands for Programmable Instruments communication protocol.
SynaccessAgent	Synaccess PDUs	In use	Monitor and control managed PDUs for remote power cycling of electronics.
UPSAgent	UPS Battery Backups	In development	Monitor state of SNMP compatible UPS battery backups.

OCS Clients

- Send commands to one or more Agents on the network
  - i.e. Servo DR to fixed temperature, take IV curve, repeat for these N temperature setpoints
- Clients take two primary forms:
  - Python script run on command line or in Jupyter
  - Javascript run in OCS-web
- Clients know the “address” for a given Agent, and make available the set of commands for interacting with an Agent:
  - start - requests the start of a Task or Process
  - wait - waits (blocking) for a Task to complete
  - stop - requests a running Process to stop
  - status - query a Task or Process’ current status, also providing access to latest data collected by Agent
- OCS Sequencer - runs clients with nice web front end for orchestrating site Agent operations (still under development)
  - Can be used in labs to run clients

OCS Network Configuration

- Network (nearly) entirely defined by small collection of configuration files (2 key ones)
  - Docker Compose File - defines docker containers running Agents
  - OCS Site Config File - defines which Agents can be run and what input parameters they should be sent at startup
- Version controlled
- DAQ expert can identify configuration issues by just viewing these files most of the time
- Allows very rapid re-standing up of a DAQ system that might need to be redeployed for whatever reason (i.e. HDD failure, etc.)
  - Clone files, minor configuration on system/installation of packages, standup system.

OCS Access Control

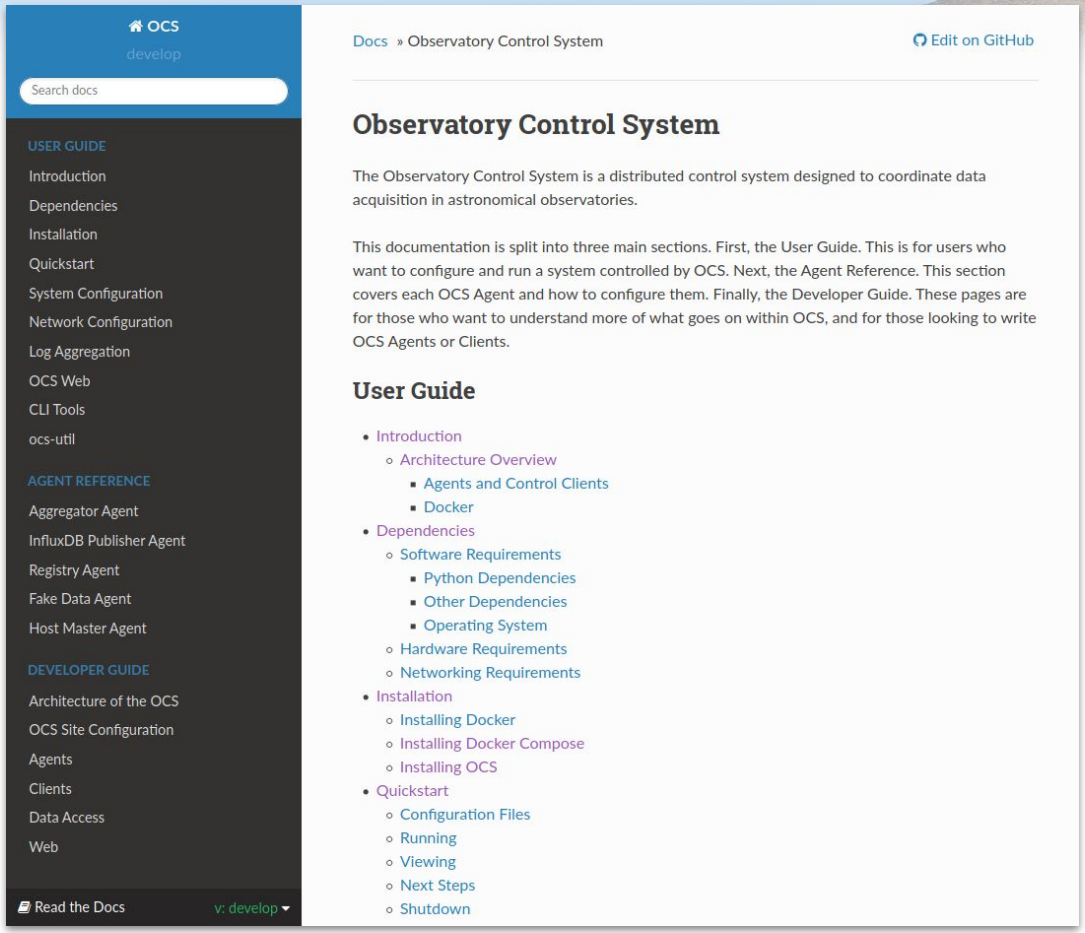
- During observations there must be some limitation on who can interact with the system to avoid unscheduled interruptions.
- The SO OCS developers are currently designing an “Access Director” Agent to enforce such limitations. (So details here still subject to change.)
  - Privilege structure with several levels, providing basic access, advanced access, and full access to Agent operations
  - Requests for temporary exclusive access to a device made to the Access Director Agent
  - Privilege level for a given client determined by credentials/token given by the Access Director
  - Level required for sending commands to a given operation defined in each Agent, and access restricted within the OCSAgent base class
- Goal is to provide some simple protection against accidental actions, and provide means to lock-down access so observatory orchestration can happen without interruption

Testing

- Core OCS code and core OCS Agents covered by set of unit tests and integration tests
- Still being expanded to cover more of the code base
- Examples of Mocked hardware connection in socs for testing Agent code w/o access to hardware.

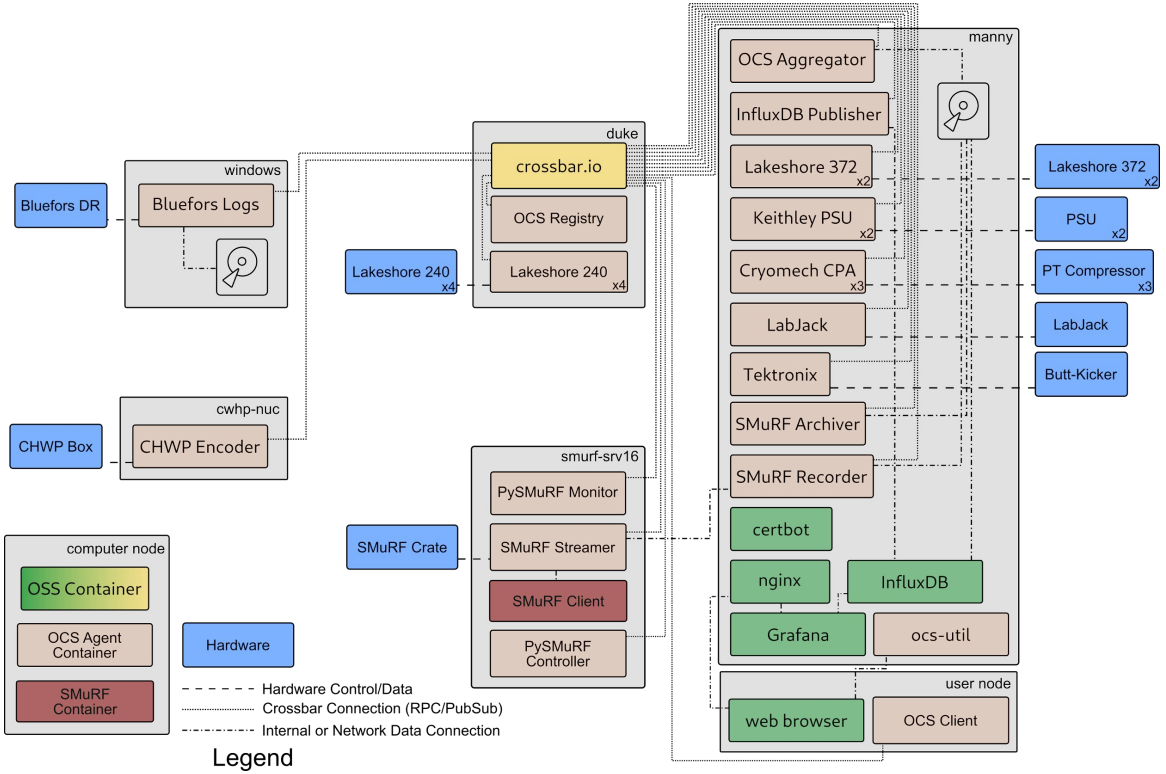
OCS Documentation

- Extensive documentation allows users to configure their own systems
- Detailed installation instructions
- Developer instructions
  - Describes Agent and client development
  - Details about various designs within OCS
- Individual Agent docs
  - Describes Agent functionality
  - Example configuration file information
- Has been key in successful adoption of OCS



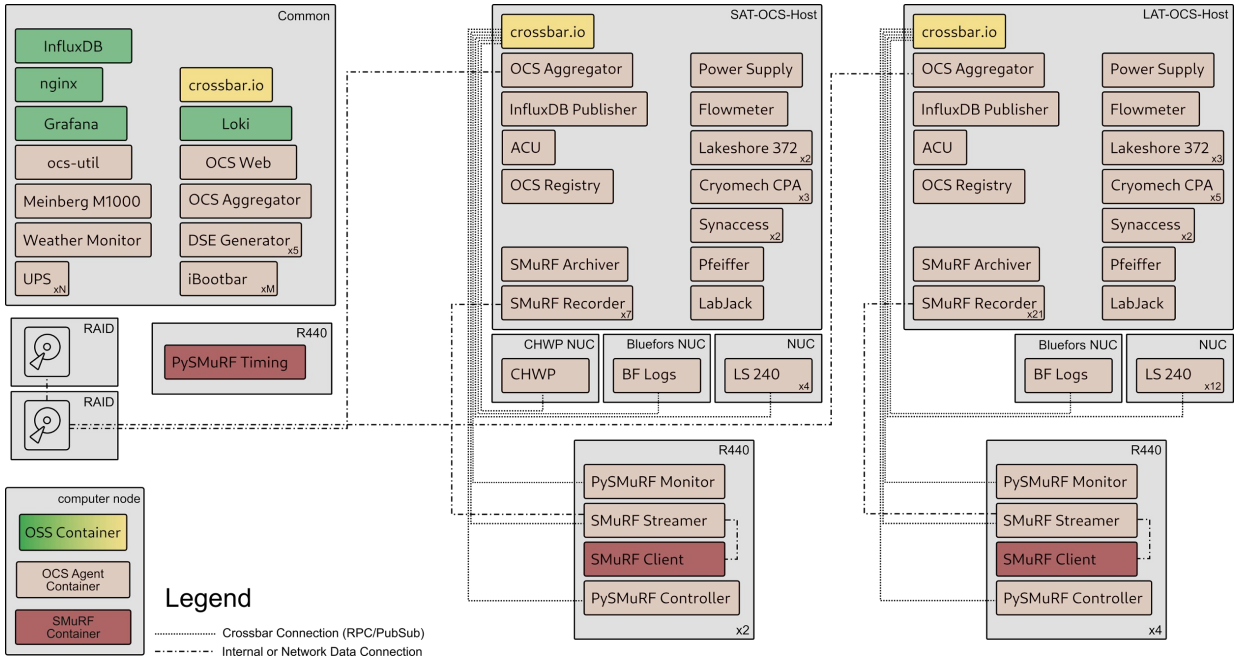
Example OCS Layout

- SAT1 deployment as of a few months ago
- Main computer (manny) runs most of the Agents, and includes live monitoring system
- smurf-srv16 runs SMuRF controller, Streamer, and Monitor agents - interfaces with SMuRF crate
- 3 other small computers for interfacing with specific hardware:
  - Bluefors DR computer (Windows)
  - CHWP NUC
  - Lakshore 240 NUC



Full Site Layout

- Full site layout is ~5x the size of the UCSD SAT1 layout
- Contains five different hosts:
  - 1x Common host - shared HK Agents, i.e. weather monitor
  - 3x SAT host
  - 1x LAT host
- Each is on its own crossbar server
- HK data published to a common InfluxDB for web monitoring
- Additional OCS hosts on SMuRF servers, hardware NUCs
- Tested at scale
- Similarly scale up for S4



Next Steps Toward CD-1

- Develop Agents for S4 hardware
- Increase testing coverage and support for mocked hardware for testing
- Scale testing OCS for CMB-S4
    - Identify any bottlenecks at large scale

Summary

- Starting from a mature design based on the Simons Observatory system.
  - Extensive documentation exists
  - Large collection of existing Agents which can be used if hardware overlaps, and referenced for development of new Agents
  - Test coverage being expanded by SO, with examples for integration testing w/o hardware access
- Ready to start developing Agents for S4 hardware.