A Look at the Past, Present and Future of CFHT's New Queued Service Observing System

CAM WIPPER & CHRIS USHER
Canada-France-Hawaii Telescope

ADASS XXXV

Kealahou

Canada-France-Hawaii Telescope has been operational for over 45 years. In that time, observational astronomy has gone through many evolutions made possible by the continued application of new technologies and methodologies – from the replacement of photographic plates with CCDs to the transition to service observing and remote operations. CFHT has been a leader in these advancements, allowing it to remain at the forefront of astronomical discovery despite the construction of progressively larger and more powerful telescopes. One of the observatory's latest evolutions is Kealahou, a reconstruction of our entire Queued Service Observing software system.

WHAT KEALAHOU DOES
"The New Way"

PI Science Proposal Submission (K1)
Structured 'Phase 1' science proposal submission interface for PIs. Includes support for concurrent calls and multi-instrument proposals.

Science Proposal Review
Full proposal review system for TACs, including review management, support for a variety of types of reviewer assignments, and in-app proposal feedback to PIs.

Proposal & Program Administration
Internal proposal and program management tools for observatory staff. Includes both consolidated overviews and proposal/program-specific interfaces.

User Management & Authentication
Self-contained user management and authentication system, including self-signup and self-serve account management tools.

PI Science Program Management (K2)
Full 'Phase 2' program management interface for science observation request submission, including CDS target query and automated finding chart creation.

Observatory Process Automation
Automatic instrument calibration sequences, time accounting calculations, and program and proposal statistics.

Telescope & Instrument Command Breaker
Generation and execution of instrument and telescope commands for all configurable observing modes, across all instruments, including SNR Goal-based and time-based observation handling.

User Collaboration Tools
Self-serve interface in both K1 (proposals) and K2 (programs) for PIs to grant access to and manage their co-Is, students, and collaborators.

Observation Scheduling Tools
Scheduling interface for querying observations and preparing nightly observation queues. Includes logbook of completed observations and QSO grading and validation interface.

Observation Execution Interface
Web-based graphical observation execution interface used by service observers.

Science Data Access
Data acquisition notification system for new program data and direct access to reduced science data through the web interface and command-line tools.

Application Documentation & PI Resources
Complete user documentation for K1 and K2, including user guides, tutorials, and instructional materials. Also provides public access to nightly observation reports and weather logs.

THE PAST
Pros, Cons, and Lessons Learned

Why redesign the QSO software system?

1. A struggling database
Over the years, as new instruments and larger science programs were added to CFHT's operations, the QSO database needed increasingly complex data models, while taking in a higher volume of data. This, in turn, put significantly more strain on the database, which began to cause technical issues during science operations as it failed to keep up with demands.

2. Usability issues
Some tools used had serious problems for users. Archaic UI design would cause confusion for PIs when filling out their program details. User accounts and science programs would have to be manually created by technical staff at the start of each semester. PIs would have to share their account password with collaborators to allow them to fill in program details. Idle user sessions would cause accounts to be locked out, requiring an admin override. Uploaded files would sometimes vanish.

3. Software maintenance
Many of the technologies used in the QSO software stack were old, outdated, and proprietary software. This posed security concerns, and some components would stop working as users updated their web browsers. In some cases, these tools had no clear upgrade path, and for others the path did exist but was clearly leading to an eventual dead-end.

4. Disconnect between related systems
In some cases, software systems that were related conceptually were totally disconnected in practice, and generally did not have a way to share data or communicate. As a prime example, the Phase 1 system for proposals was completely detached from the Phase 2 system for programs. This resulted in data needing to be manually tracked and re-entered by both CFHT staff and PIs.

5. Poor programmatic control
In the modern era of astronomy with the widespread adoption of Python, many users are more code-savvy, leading to a wider demand for the ability to perform programmatic data entry to manage programs with more complex observing strategies.

Looking forward, a programmatic way to access data was also of interest for developing internal operation tools for increased observatory automation.

Development Strategy

Old and new, operating in parallel

When development on Kealahou began, it was decided that it would run in parallel with CFHT's legacy QSO applications and databases. Data would be shared between the systems by having Kealahou write updates back to the legacy database, and by regularly running a database "syncer" task to copy data from the legacy database into Kealahou. Initially, Kealahou would be developed to run SPIRou, CFHT's latest incoming (at the time) instrument, and existing instruments would later be ported over one by one.

Pros

Reduced risk of seriously disrupting operations with a bug or issue in the new system

Able to incrementally make use of improvements added by Kealahou

Gradually decreased strain on old system as each instrument was migrated

Cons

System split by instruments has caused confusion for PIs (and even staff)

Development effort spent on ensuring compatibility has scaled with size of glue code, test code, and the overall code base

Underestimated spiking performance impact of syncing between systems

A broadly scoped software project

The Kealahou project was scoped to encompass a broad array of software responsibilities for the observatory, including functionality for both internal and external users. While not strictly necessary to fulfill these requirements, this naturally led the software components of Kealahou to have a similarly monolithic design.

Pros

Able to more quickly develop a broad set of features for rapid iteration early in the project's life

Less work to make systems which are not closely related interact with each other

Less effort to maintain functional compatibility between subsystems

Easier to create a cohesive user experience

Cons

Upfront work to support broad requirements early on at odds with further development as requirements shift

Difficult to on-board developers to work on a subset of project

More effort to maintain unrelated subsystem code when refactoring

Unable to take down just part of the system for maintenance

Do you have comments or questions about Kealahou -- on how the application works, or the technologies and techniques used by the team?

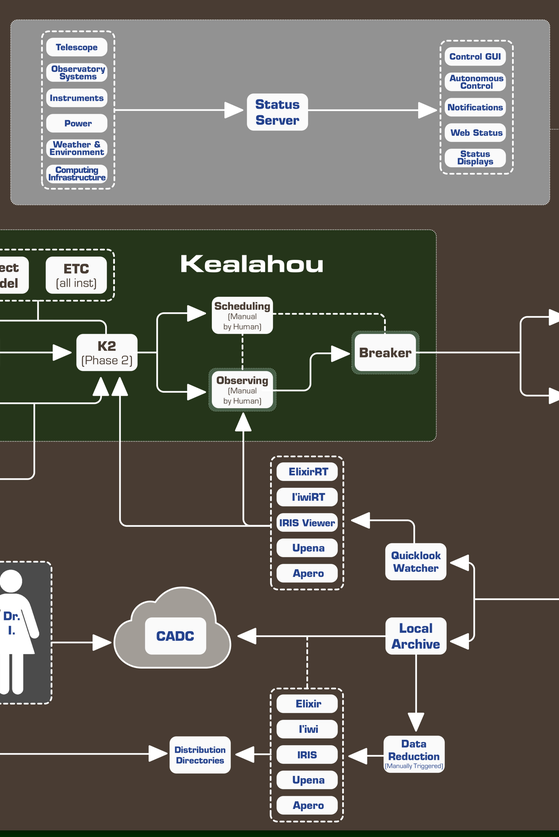
If you do, we want to hear from you!

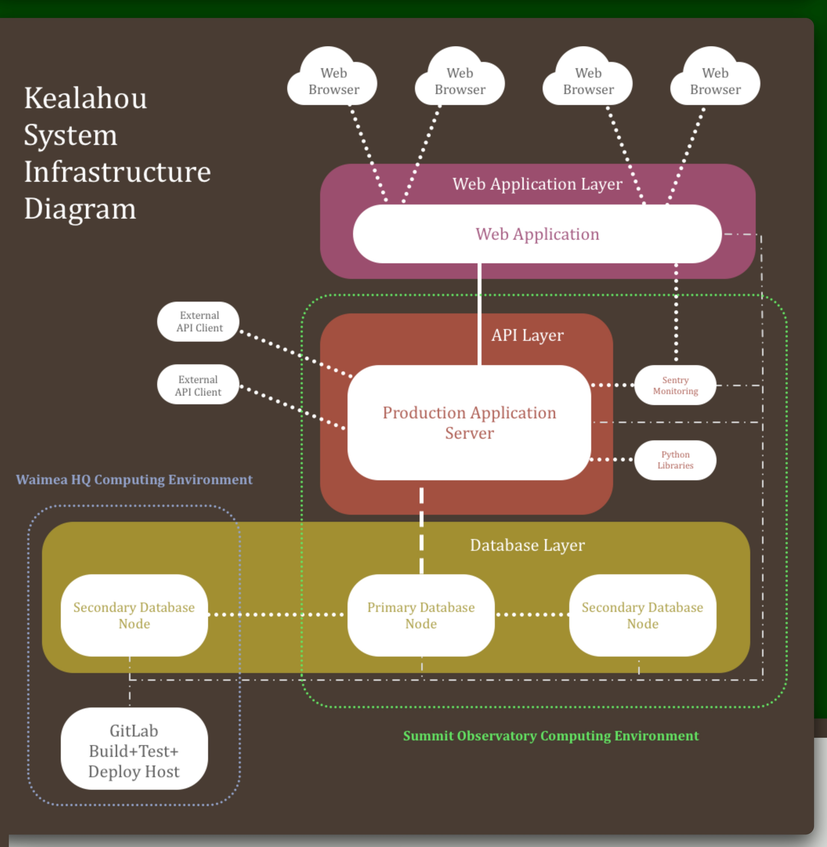
Contact the authors via email:

Cam Wipper: wipper@cfht.hawaii.edu // Chris Usher: usher@cfht.hawaii.edu

Additionally, the Kealahou Team would like to acknowledge CFHT's privileged location on Maunakea, a significant historic and cultural site to the indigenous Hawaiian people. The Kealahou Team conducts all development with the utmost care and respect for Maunakea.

CFHT Observing Computing Infrastructure & Science Data Flow





THE PRESENT
A Custom Platform for the Modern Era

The Database

The backbone of the QSO software system is its database. For Kealahou, we replaced the Database Management System (DBMS) software, as well as revamped the data models to improve database performance and give the system more flexibility going forward. One particular change we made was utilizing data "blob" fields for each entity rather than storing all data directly in columns. Beside the performance improvement, this provided a convenient way to store data with flexible configurations.

Legacy QSO DBMS

Sybase ASE 12.5

Proprietary, closed-source software

Difficult to find information without commercial support

Released in 2001, EOL since 2009

Newest ASE major version released in 2014, no clear upgrade path

Kealahou DBMS

MariaDB 10

Open-source software

Information widely available online

Released in 2014, LTS until 2028

Upgrade path to latest LTS version gives support till 2033

The Web Interface

The top layer of the Kealahou system is the web interface. This web application provides external users a single unified UI to access Kealahou, as well as giving CFHT science staff high-level management tools.

Legacy Phase 1 & 2

Phase 1 proposal submission and review through NorthStar application developed by Astron since 2010

No access to NorthStar source code for advanced changes. Support requires knowledge of several different tools and technologies

Separate web tools for Phase 2, Program Viewer, Night Reports, called PH2

PH2 is an internally developed tool built in ColdFusion 5 (2001)

All applications require specialized software on web server

Features in both applications break with web browser updates

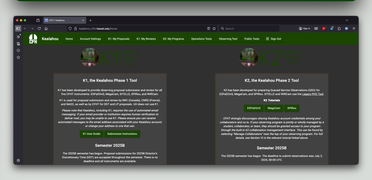
Kealahou Phase 1 & 2

Access to all stages of the QSO process through a unified web application

Built in TypeScript and Angular

Served as basic HTML + CSS + JavaScript

Works on modern web browsers



The API

At the core of Kealahou lies a unified web API that the web interface and other client applications utilize to access the Kealahou database, as well as to trigger actions for observatory operations.

API specification using Protocol Buffers
Application server in Java

REST-like endpoints using JSON over HTTP
Alternative gRPC protocol used by internal tools

Legacy QSO application logic structure

Database stored procedures used for system logic, tying critical rules to the DBMS

Fetch and persist data with database queries

Collation of data from related entities generally must be performed in front-end application logic

Necessary database post-processing steps in separate programs/scripts written in C, Perl, Python, Bash

Kealahou application logic structure

No business logic stored in database (beyond storing entity relations)

Fetch and persist data through a web API

Data returned by the API generally already represents complete entity data – just needs filtering/formatting as desired

Core functionality is baked into the API services, additional tools communicate through the API

AEON & rToO

Starting in 2026, CFHT users will be able to submit targets and observation requests to Kealahou from the AEON network through the Kealahou API. This will include support for both fixed and moving targets. Additionally, this project is the implementation of a rapid Target of Opportunity (rToO) capability, cutting ToO observation delays from 'hours-to-days' to 'seconds-to-minutes'.

As part of this project, we will publish a full API specification, opening up new opportunities for users to interact with CFHT in a programmatic way, either directly, or through their own software tools and systems.

Beyond

Following this, we have some long-term ideas for Kealahou as well. This includes automatic queue scheduling, autonomous observing, integration of natural language processing, and spinning off K1 or other parts of Kealahou into open-source modules.

Kealahou is expected to serve CFHT well into the next decade.

THE FUTURE
Kealahou In the Age of LSST and AI