HALCYON: Unified HPC Center Operations

Under a shared campus cluster model, with many research groups investing distinct amounts and annual new hardware acquisitions, management of users and resources can become very complex. At Purdue, the Research Computing division set out in 2011 to design a cluster management solution to empower faculty to manage access to their own purchased resources.

As operations at Purdue expanded, the internal portal evolved and took on many aspects of the operation of an HPC center beyond resource allocation and management. Eventually, components included HPC and storage resource management, user management & authorization, customer relationship, communications, documentation, and purchasing.

Halcyon reconstitutes these in a modular, extensible framework to allow for the growth and maintenance necessary to encompass all aspects of HPC center management, from resource allocation to the less talked about but equally important aspects of customer relations. This will allow centers to operate not only more efficiently, but more effectively, and deliver better services to researchers.

A key aspect to the interaction of external resources such as compute clusters or storage is the use of an internal message queue, implemented through ZeroMQ. When users submit changes such as new directories, permissions changes to a directory, or checks for current resource usage, a message is queued for a remote worker on the appropriate resource. The worker processes this message as soon as possible and submits any response back to Halcyon asynchronously.

Solution

The original Purdue Research Computing portal served its purpose well over the years but difficulty in making changes and the expanding needs of managing an HPC center drove the decision to re-architect the underlying code to:

- Abstract all environment assumptions and settings
  - Language and site settings pulled into internal database and interfaces for editing
  - Environment settings incorporated and consolidated into configuration files
- Restructure for modularity and easier extension
  - Laravel as an underlying framework due to its popularity, large community, active development, and inclusion of desired features
  - Laravel app design helps secure codebase

The portal code was separated into modules—logical groupings based on the data being handled, tasks, and interfaces. Modules may communicate and share data via an event dispatcher. Third-party services are integrated as plug-ins through the same event dispatcher system, which allows modules to ingest data from sources such as LDAP or REST APIs, as shown in Fig. 1.

Modules implement a REST API documented in the code according to the OpenAPI v3 specification, allowing automatically generated human-readable documentation for all available endpoints. This API is one of the essential interfaces for external services and resources. For administrative and scheduled tasks, modules may also implement console commands, providing powerful methods to inspect and interact with data. Some of these interfaces are shown in Fig. 2, Fig. 3, and Fig. 4.

Results

In its initial incarnation, the Purdue Research Computing portal provided faculty, students, and center staff an immediate reduction in time and effort to request, provision, and manage allocations. The user allocation/authorization alone was estimated to save 6.7 hours of center staff time per week over the first 18 months. New storage spaces were able to reduce time-to-completion from 8 days to 1 hour within the first year of adoption of storage management. Pushing those tools beyond the resource to larger center operations has proven still more valuable, but all this originally came at a high maintenance cost.

Reconstituting the portal features into a modular, extensible framework was an essential next step to allow for the growth and maintenance necessary. These aspects of Halcyon have already allowed improvements to user interfaces, workflows, and integrations to proceed at an even-increasing pace. To date, Halcyon is gaining praise from both center staff and researchers for time savings in additional self-service utilities and automation of repeated tasks.

Future Work

While there has been significant re-architecture in the creation of Halcyon to increase modularity and extensibility and to remove hard-coded values, there still remain a few values and interactions built into code which are specific to the environments and implementations of Research Computing at Purdue and which must be removed.

Implementing Halcyon at another organization is a current goal, which will further highlight where flexibility and configuration can be introduced to better accommodate different infrastructure and HPC center policies. This will provide an opportunity to develop more integrations and plugins to further extend Halcyon’s capabilities.

References


