HIL and QUADS Project Proposal

Vision and Goals Of The Project

Design and implement an integration solution that allows QUADS to interact with and schedule nodes allocated by HIL. It will allow users to schedule and perform large scale experiments on secure, isolated physical hardware in the MOC.

As it currently stands, HIL (Hardware Isolation Layer), provides the allocation, isolation, and connection of physical (bare-metal) hardware and networks in the MOC. It is meant to provide a minimal layer of abstraction between the physical hardware makeup of a datacenter (whose node and switch interfaces can differ across a diverse hardware layout) and the node provisioning services which benefit from a common interface and do not need to be exposed to the bare metal. HIL's scope is restricted to allowing hardware allocation and isolation, but does not provide any rules for how nodes and switches are chosen, and when they are allocated for use.

QUADS (Quick and Dirty Scheduler) provides automated scheduling and time-based allocation for hardware nodes. While QUADS also provides similar hardware allocation as part of its scheduling, it currently is hardwired to work only with specific hardware (Juniper), and cannot provide its service as decoupled from the hardware layer.

Our project would allow QUADS to rest on the layer of abstraction provided by HIL, and could decouple it from its dependence on a specific hardware. It would allow MOC users to take advantage of the security and portability of the HIL, while no longer having to manually schedule allocation and usage, which could instead be automated and maintained by QUADS.

Users/Personas Of The Project:

The services provided by the integration of HIL and QUADS will be used by researchers and students from MIT, Harvard, BU, NU and UMass, as well as Open Source developers and the industry partners of the MOC.

It targets MOC users who use HIL to allocate physical hardware, and want to utilize a scheduling application for the isolated nodes.

It does not target MOC users with small scale projects not in need of automated resource management.

1. Scope and Features Of The Project

- Analysis of HIL and QUADS codebases
    - Expose potential connection and extension areas where the applications could be easily integrated
    - Establish data necessary for each application to function, and how each application stores and structures that data (e.g. YAML, JSON, etc)
- Extension of the APIs of each service to allow compatibility, or creation of a new API
- Possible implementation of driver to extend HIL's functionality to Juniper nodes/switches (this is not an immediate goal of the project, but could be a stretch extension that would allow current users of QUADS to use HIL, instead of just HIL being able to use QUADS)

Out of Scope:

- Providing any new functionality within either application aside from simple integration (i.e. integration should not compromise any original functionality, and the applications should be minimally changed)
- Adding any scheduling or allocation logic (as stated above, the two services should remain as before, but will now be compatible with each other - project is to integrate, not innovate).

2. Solution Concept

Global Architectural Structure Of the Project:

The architecture of the project involves extending each service so they are compatible. It may involve:

- Modifying hooks within the two services so they can communicate via their existing APIs.
- Designing and implementing a new API to enable integration, although the level of complexity and functionality of the API is not yet clear.
- Ensuring no change in original functionality of each application (users will still be able to use either service without the other - applications will remain loosely-coupled)

The first phase of the project consists of information gathering and testing the functionality and compatibility of the applications. As we acquire more information about HIL and QUADS, we will better understand the extent to which we will need to modify or extend each service, which will allow us to identify the best-suited solution.

The HIL diagram below is from an article provided on its project page "HIL: Designing and Exokernel for the Datacenter" (http://dl.acm.org/citation.cfm?doid=2987550.2987588), and the QUADS diagram is from its documentation in its github repository (https://github.com/redhat-performance/quads).

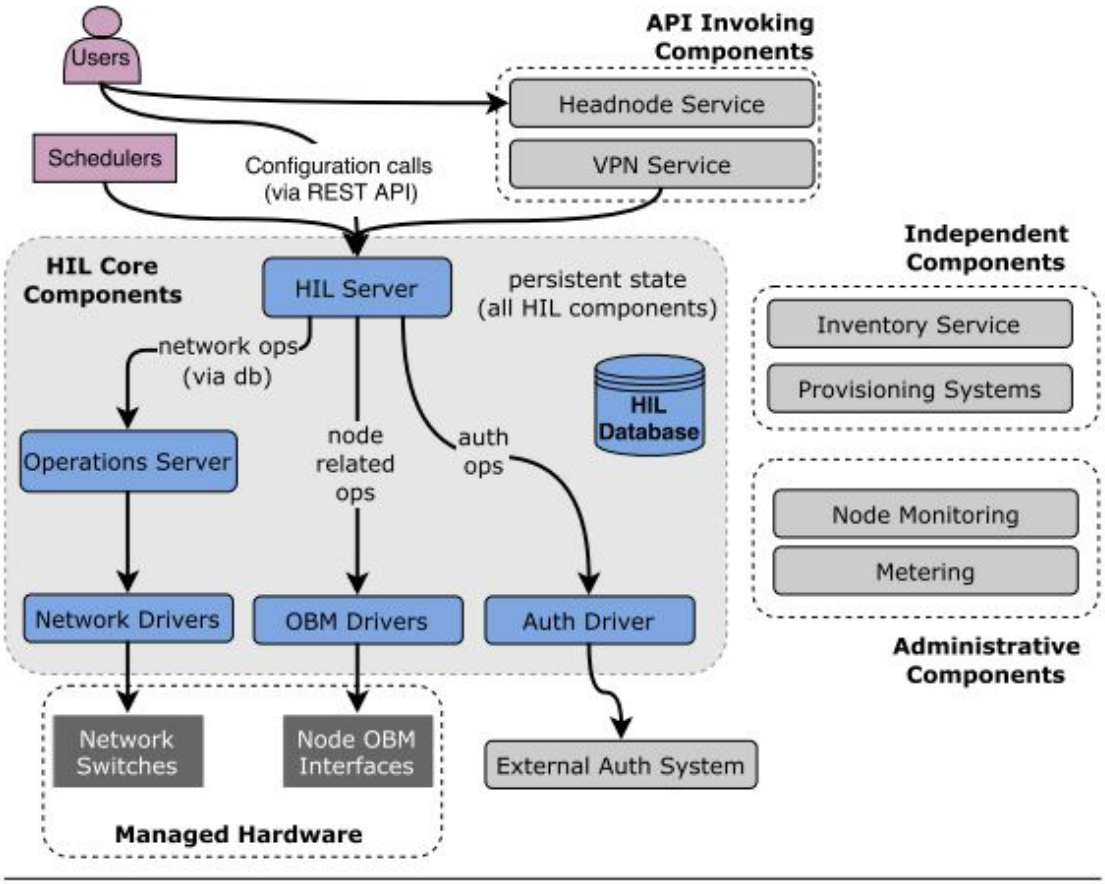
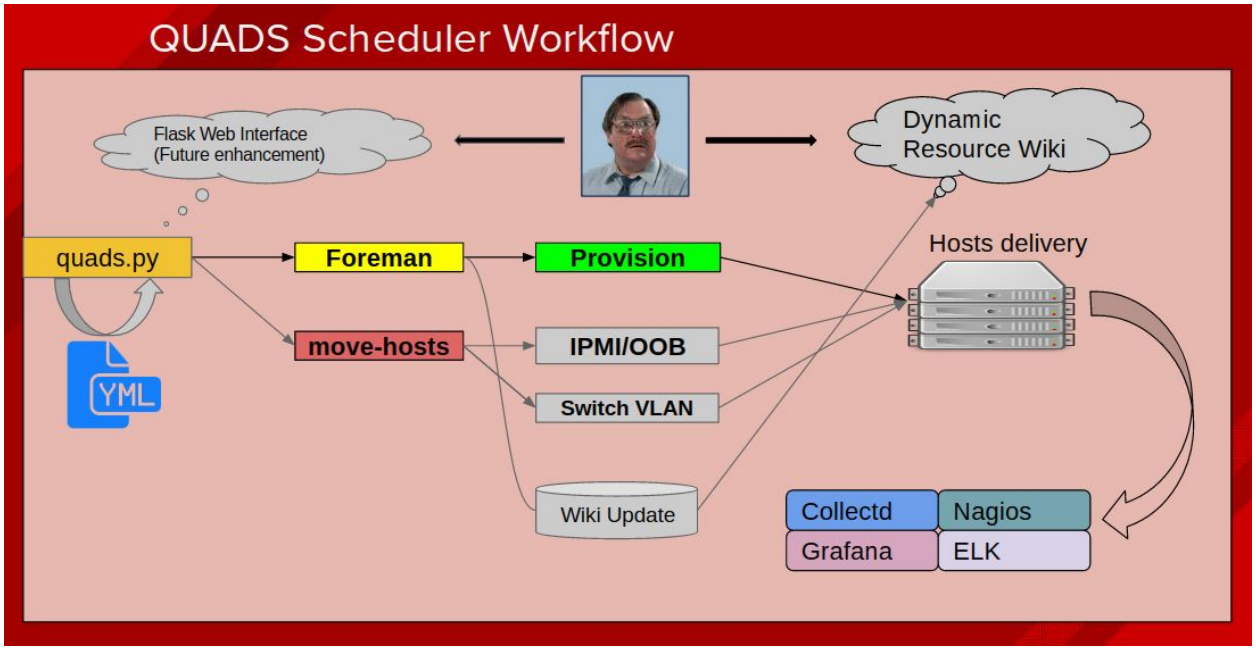


Figure 2. Components in a HIL deployment.



The high-level application architectures above indicate areas where the services could integrate. The HIL diagram even includes an entrypoint for schedulers, which will serve as a guide when further analyzing the codebase. From our initial research and experimentation with the code, the connection point from the perspective of the QUADS diagram may lie somewhere within the "move-hosts" segment. HIL provides the ability to safely transfer resources between hardware nodes, and uses IPMI and VLAN to control power-cycles and networking respectively. HIL also provides coupling with external provisioning services (such as Foreman, used by QUADS), which also provides an area of overlap that necessitates further investigation.

Design Implications and Discussion:

As the first deliverable of the project is the results of our codebase analysis and testing, there remain a few potential design options for the project, which cannot be appropriately chosen without the analysis provided by the first sprint. The key points of design-related discussion currently are:

1) Integration solution design - As both services already expose a RESTful API, it may not be necessary to make significant modifications to either codebase. Design options include making minimal changes to the HIL API, the QUADS API, or both to allow for transparent data exchange between them. However, if it becomes evident through our investigation that the way each application stores and shares data is incompatibly different, it may be necessary to implement a new API that sits in the middle and reorganizes the necessary

data, providing a common interface for easy communication between applications.

2) Use cases - Will an end user begin using our integration from the HIL side, QUADS side, or both? As both QUADS and HIL support a CLI and currently require some data to be entered manually, which application be modified to automatically supply the other with the required data? Will our end goal allow the capacity for both? From the project description, our research, and discussions with our mentors, we are only required to create the capacity for one, but a stretch goal will be to have them work both ways.

Acceptance criteria

- Functional RESTful interface calls between HIL and QUADS applications
- QUADS can schedule and allocate hardware nodes and switches controlled by HIL
- HIL and QUADS continue to function independently and remain loosely-coupled (i.e. HIL users still have the option to schedule manually, or integrate another scheduling service, etc)
- HIL is extended to manage Juniper switches, allowing all current QUADS users to be able to use HIL if desired (stretch goal)

Release Planning:

As much of our design plan will be based on the outcome of our first sprint, we have sketched out a tentative plan of the next sprints that will most likely be updated as we progress.

Sprint 1: February 23

- Use testing environments and processes of both HIL and QUADS as a starting point for gathering information on where they overlap and where they differ - base all questions on the success or failure of unit tests
- Isolate points of connection and points of conflict between applications, paying attention to data and timing (an initial goal of the project is to fill in these gaps), outline where layers of abstraction meet
- Present and organize information gathered about the functionality overlap between HIL and QUADS, and where there are gaps we need to fill in. Demo passing and failing tests of both applications, and display the areas where we will need to build on/modify in order to integrate

Sprint 2: March 16

- Based on conclusions from the first demo, select appropriate design option for the integration
- Finalize high-level architecture of implementation and APIs (either modified or created)
- Demo existing / new APIs that we will need to use

Sprint 3: March 30

- Demo at least one functional interaction between applications (e.g. QUADS can request information about nodes from HIL, but cannot yet schedule them)

Sprint 4: April 13

- QUADS can perform all basic services interacting with HIL
- Demo successful use cases

Final Demo: April 27

- Integration fully implemented and tested
- HIL has been extended to function on hardware QUADS is hardcoded to utilize (Juniper switches), allowing for complete functionality of QUADS on HIL (stretch goal)