Recursive HaaS and Fast Provisioning

1. Vision and Goals Of The Project:

By lowering the barrier of entry, **Recursive HaaS** will allow any customer to become a utility computing provider. It will provide users with the freedom to try new solutions in the domains of utility computing and resource provisioning without the need to secure enormous capital investments to procure the necessary infrastructure. Recursive HaaS has the potential to not only provide novel models to trade hardware resources but also to make it easy to add new features to HaaS itself without affecting the stable deployment. A key feature of Recursive HaaS is the ability for **Fast Provisioning** where operating systems are deployed as fast as possible onto HaaS nodes allocated to customers.

High level goals of the project are as follows:

- Create a layer of HaaS that will run over existing HaaS
- Provide the ability to provision resources from HaaS over which a third party can host their own version of HaaS
- Ability to allocate nodes and other services just like base HaaS
- Ability to add new features to recursive HaaS without affecting base HaaS
- Adding fast provisioning of nodes to test the ability of recursive Haas to incorporate new features

With fast provisioning the high level goals are as follows:

- Find a fast and efficient way to boot an arbitrary number of nodes with an OS of client's choice
- Provide a solution to return the nodes to a consistent state into the free pool without rebooting, thereby reducing time-lag in node provisioning

1.1. Users/Personas Of The Project:

Recursive HaaS will be used by:

- Government agencies to host their private clouds
- Hospitals to perform computationally intensive operations
- Entrepreneurs to resell HaaS services
- The open-source community to safely test HaaS innovations

2. Scope and Features Of The Project:

rHaaS: Recursive HaaS
bHaaS: Base HaaS

In Scope:

- Build command line utilities for users to request resources from rHaaS
- Construct RESTful APIs to provide resource provisioning using the web
- Build back-end support for communicating with bHaaS for deploying configurations requested by customers of rHaaS
- Create usage documentation for administrators
- Create documentation to be used by developers and contributors of code
- Provide a platform to efficiently push OS images of customer's choice to all their nodes in a fast and efficient way
- Provide a solution to reclaim nodes from customers to the free pool of rHaaS without rebooting

Out of Scope:

- GUI-based solution for rHaaS
- Scalability of the HaaS implementation
- Security (recursive HaaS is equally secure as HaaS, and equally vulnerable)
- Including support for fast provisioning any version of Windows or Mac OS

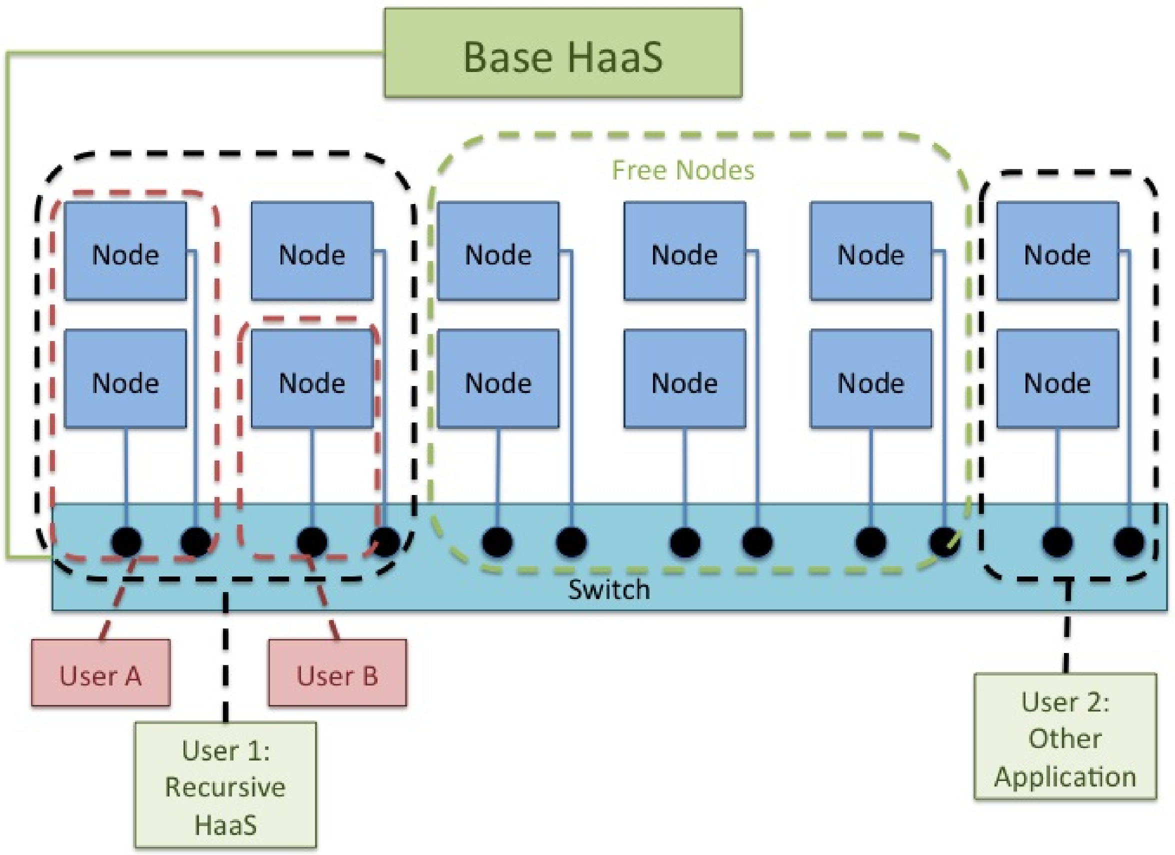
THIS SPACE INTENTIONALLY LEFT BLANK

3. Solution Concept

3.1 Global Architectural Structure Of the Project:

3.1.1 Recursive HaaS

The following architectural diagram shows a small-scale representation of a base HaaS with twelve server nodes and two user instances, one of which is running recursive HaaS.

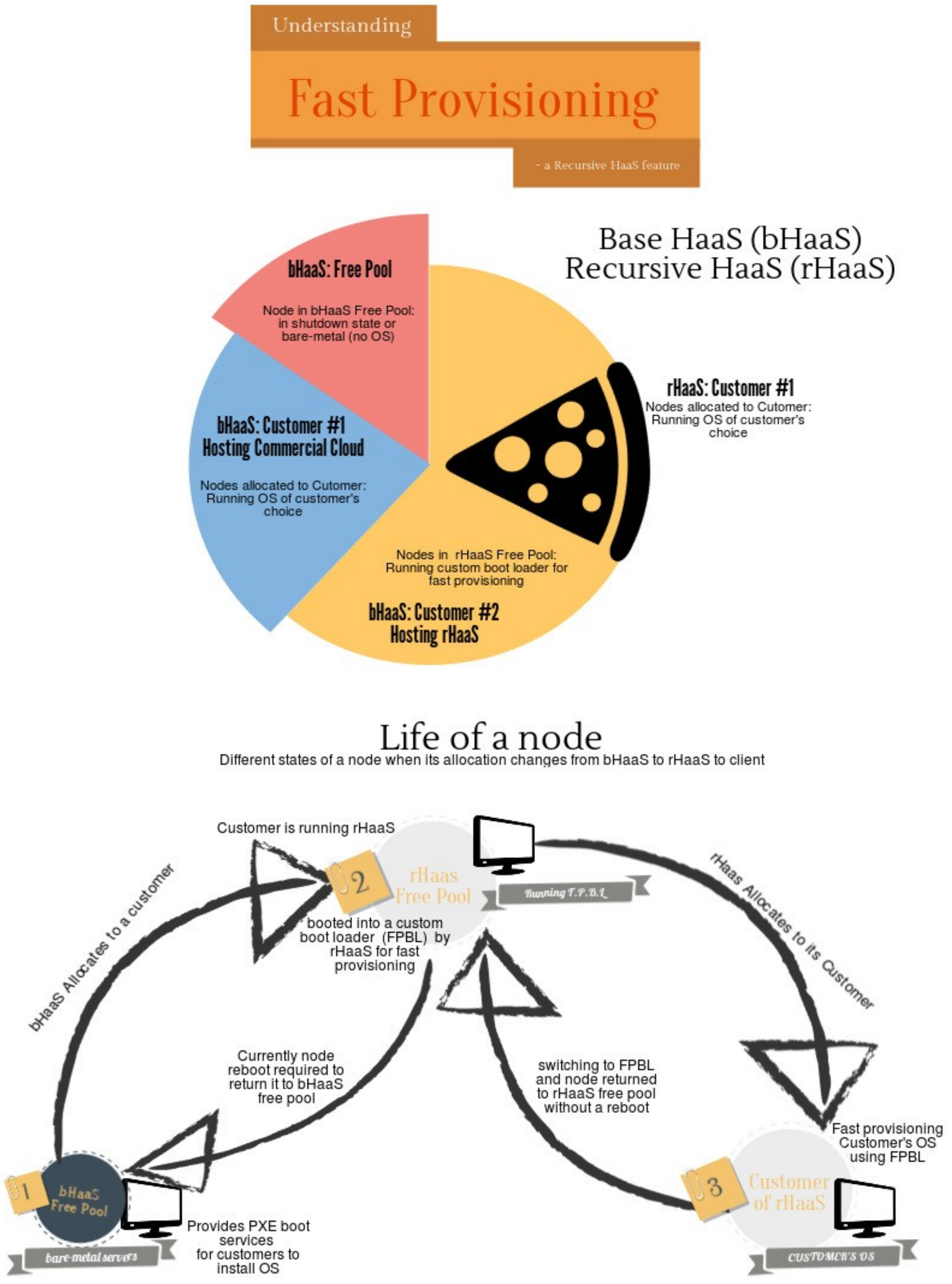


The diagram illustrates a collection of twelve servers managed by a base HaaS installation. Four servers have been provisioned for User 1, and two for User 2. The remaining six are in the free-pool, waiting to be assigned as more resources are requested by new or existing users.

In this diagram, User 1 is running an instance of recursive HaaS. Recursive HaaS further subdivides these four servers into a two-server instance for User A, a single server for User B, and a single free node. In this instance, User 1 can manage the single free node, request more resources from the base HaaS as needed, or release nodes back into the base HaaS free-pool if they are no longer in use.

3.1.2 Fast Provisioning:

The following diagram demonstrates how rHaaS, bHaaS and Fast Provisioning are related. It also provides an overview of how Fast Provisioning affects any node during its provisioning life-cycle.


Understanding
Fast Provisioning
- a Recursive HaaS feature
Base HaaS (bHaaS)
Recursive HaaS (rHaaS)
bHaaS: Free Pool
Node in bHaaS Free Pool: in shutdown state or bare-metal (no OS)
rHaaS: Customer #1
Nodes allocated to Cutomer: Running OS of customer's choice
bHaaS: Customer #1 Hosting Commercial Cloud
Nodes allocated to Cutomer: Running OS of customer's choice
Nodes in rHaaS Free Pool: Running custom boot loader for fast provisioning
bHaaS: Customer #2 Hosting rHaaS
Life of a node
Different states of a node when its allocation changes from bHaaS to rHaaS to client
Customer is running rHaaS
2 rHaas Free Pool
Running F.P.B.L
booted into a custom boot loader (FPBL) by rHaaS for fast provisioning
bHaaS Allocates to a customer
rHaas Allocates to its Customer
Currently node reboot required to return it to bHaaS free pool
switching to FPBL and node returned to rHaaS free pool without a reboot
Fast provisioning Customer's OS using FPBL
1 bHaaS Free Pool
bare-metal servers
Provides PXE boot services for customers to install OS
3 Customer of rHaaS
CUSTOMER'S OS

In their initial state, all nodes belong to the free pool of base HaaS. At this stage they do not have any OS installed - in other words, they are bare-metal servers.
A customer can install any OS of their choice via PXE, booting the bare-metal server procured from the free-pool of bHaaS.

rHaaS also uses the PXE booting feature and installs a bootloader on each server it procures from base HaaS. We will call this bootloader Fast Provisioning Boot Loader or FPBL. Unlike the free-pool of bHaaS, the default state of all the servers in the free-pool of recursive HaaS is to be booted into FPBL and ready to be allocated.

When a customer requests nodes from recursive HaaS, we will provide a way to install all the allocated nodes with customer’s OS as fast as possible.

When the customers is done using the nodes and wishes to return them to the free pool, we will employ the fast provisioning method to switch back to FPBL from the customer’s OS without rebooting the nodes.

3.2 Design Implications and Discussion:

The global architecture designs were made during a team meeting and verified by our project mentor. They are based on our current understanding of the HaaS topology. As research is conducted and mentors are consulted, further design decisions will be made.
The following points support the design decisions made so far.

- Jobs regarding network configuration deployment and isolating resources from free-pool will be forwarded to bHaaS from rHaas. This will avoid conflict between two HaaS servers trying to manage the hardware resources in the same pool. Also, this will reduce latency in realizing the changes requested
- We will make the RESTful API of recursive HaaS the same as that of base HaaS to maintain consistency in design
- For Fast Provisioning, our potential design is to boot the nodes into a kernel which will later switch to the OS provided by the customer. Similarly when customers wish to return the node, we will devise some mechanism by which nodes can switch back to the bootloader of rHaaS, wiping the previous image
- We believe having nodes booted into a custom image will save time and aid in the fast provisioning of servers. Reclaiming the servers and bringing them to their default state without rebooting will also save time otherwise lost to booting the system

Research topics we will explore in support of project goals include:

- Choose a bootloader for fast provisioning
- Other Recursive HaaS-like solutions
- Existing pre-boot environments
- Kexec utility and alternatives
- Solutions/tools that switch a system to a new OS without rebooting
- OS selection to implement fast provisioning

4. Acceptance criteria

Following is the minimum acceptance criteria

- Being able to provision two servers from the resource pool of rHaaS to a customer
- Provision the customer's image of an OS of the customer's choice onto the nodes
- Reclaim the nodes, wipe of the customer's image of OS without rebooting the nodes
- Reassign the reclaimed nodes to another customer without rebooting the nodes

5. Release Planning:

The latest detailed user stories and planning are available on the Trello Board.
https://trello.com/b/9kxWc4hw/recursive-haas-with-fast-provisioning

Release #1 (due by week 6):

- User stories: HaaS Reseller node provisioning
  - Using the CLI, the User provisions nodes from base HaaS
  - Using the CLI, the User provisions nodes using recursive HaaS
- TODO

Release #2 (due by week 8):

- User stories: HaaS Reseller queries
  - Using the CLI, the user queries to gather information about provisioned recursive HaaS nodes
  - Using the CLI, the user queries to test state of released recursive HaaS nodes
- TODO

Release #3 (due by week 10):

- User stories: HaaS Reseller forward node provisioning
  - If the user requests a node that exceeds recursive HaaS freel pool capacity, the reseller can optionally automatically request more nodes from base HaaS
- TODO

Release #4 (due by week 12):

- User stories: Haas Reseller releasing nodes
    - If the user releases nodes, they are returned to the recursive HaaS free-pool or to the next sublayer of HaaS/ recursive HaaS
    - Released nodes are returned to bootloader state without rebooting
- TODO

Release #5 (due by week 14):

- TODO