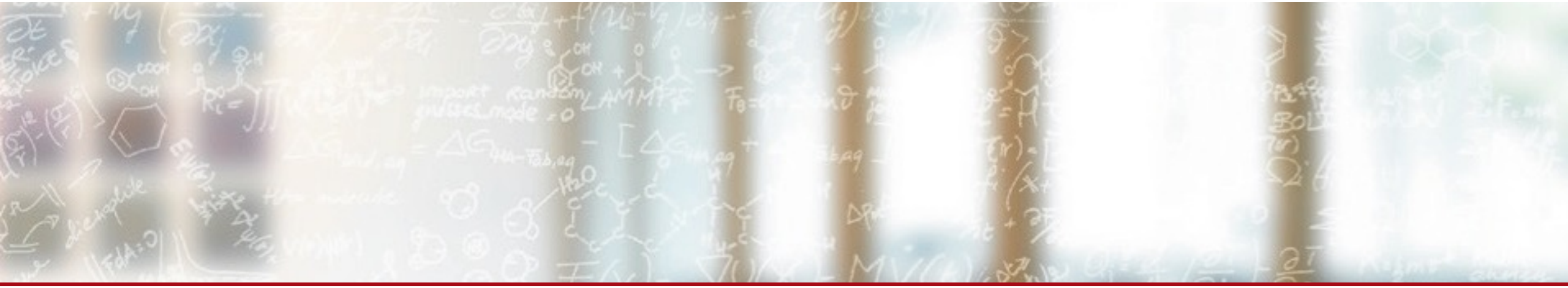
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

ETH zürich



Experiences Deploying Slurm Clusters on OpenStack using Magic Castle

Victor Holanda Rusu, CSCS

May 19th, 2022

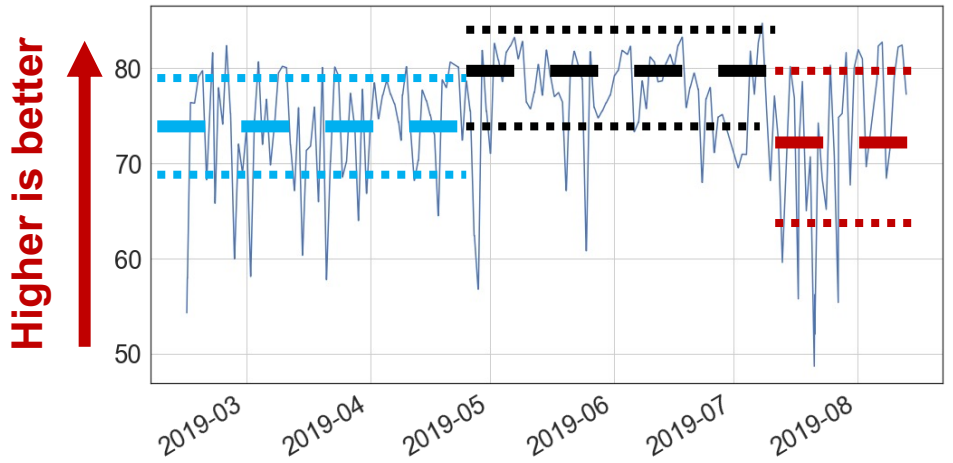
Before we go into the topic, a bit of history and context…

User Experience Monitoring at CSCS

CSCS tools in action

ReFrame

- 2017 - CSCS released, **ReFrame**
  - Can be used to test any system, especially great for HPC environment testing
  - Used to implement (some) service level indicators (SLI) in production
- 2019 - at HPC.ch, talked about monitoring tool
  - Named **Performance Monitor**
  - Can be used track service level expectations (SLE)
  - We could distinguish performance changes based on average and variation independently

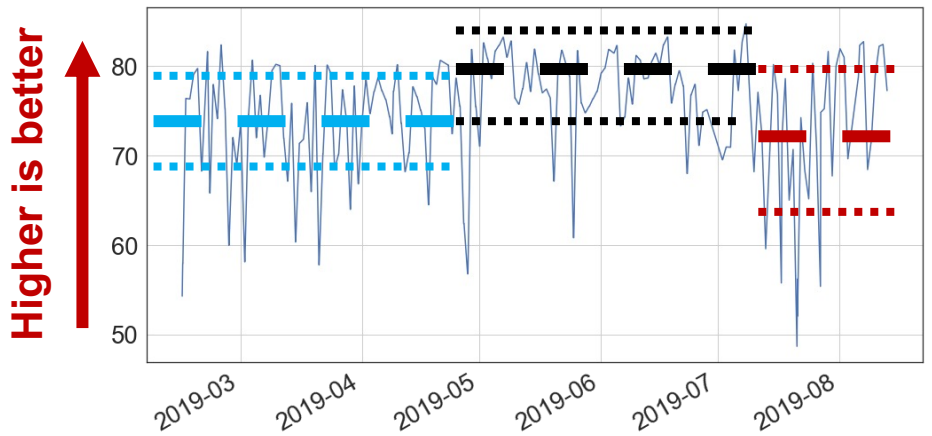


We don't want to "just" observe, we want also to know the causes

Can we isolate the root causes of the performance changes?

Can we understand why an application suffers from slowdows in production?

- There are two ways of studying root cause of performance changes
  - Formal Methods (Mathematical proves)
    - The “right” way of doing it, but difficult to do
    - In general, requires an abstract view of the system and its metrics and then one needs to right a code that represents the abstract model
  - Probabilistic Methods (No proves, just “demonstrations”)
    - Need to have enough (a lot of) statistical data
    - We can only find high probability that events are correlated



Can we isolate the root causes of the performance changes?

The playground

- We cannot be experimenting with Piz Daint
  - Can we apply a QoS so that all network traffic is 20% slower?
  - Can we throttle the scratch file system or the CPUs?
- At CSCS we have a OpenStack cluster
  - We have CPU overprovisioning
  - Good ground for testing cluster isolation
- Magic Castle
  - Ease of use
  - Meant to be also used by non sys admins
  - Contains an out-of-the-box software stack
  - Enables SELinux

What is Magic Castle?

What is Magic Castle?

The HPC environment creator

- Aims to recreate Compute Canada's user experience in public clouds
- Code can be dowloaded from https://github.com/ComputeCanada/magic_castle
- Uses HashiCorp Terraform (https://www.terraform.io) to define the infrastructure
- Uses cloud-init (https://cloudinit.readthedocs.io/en/latest/) to perform the initial virtual machine configuration (boostrap Puppet)
- The cluster configuration is performed using Puppet (https://puppet.com) and HashiCorp consul (https://www.consul.io)
- Compatible with AWS, Microsoft Azure, Google Cloud, OpenStack, and OVH
- Initially used for teaching and testing of HPC technologies

Magic Castle's provisioning objectives

The "Magic" part

- Has to be automated (no human intervention)
  - Just configuration files required to define the cluster
- Provision master, login, and compute nodes asynchronously
- The cluster needs to stay healthy and secure by itself
  - When one automates cluster creation, creating clusters becomes "too easy" (more clusters than sysadmins to manage)
  - The end user may not even be a sys admin
  - Comes with SELinux enabled with user confinement, also in the compute nodes

What is Magic Castle?

The “Castle” it builds

- Minimally requires 3 instances (Management, Login, and Compute)
- Basic configuration requires 3 mount points (/home, /scratch, and /project)
- Full software stack from
  - Compute Canada
  - EESSI (https://eessi.github.io/docs/)
- Batteries Included
  - Slurm scheduler (https://slurm.schedmd.com),
  - NFS,
  - Globus Endpoint (https://www.globus.org),
  - JupyterHub with Batch Spawner (https://jupyter.org/hub),
  - FreeIPA with Kerberos, Bind,
  - 389 DS LDAP (https://www.freeipa.org), and
  - Lmod (http://lmod.readthedocs.io).

How Magic Castle uses its four configuration management tools?

HashiCorp Terraform

Infrastructure provisioning tool

- Terraform supports multiple providers (AWS, Google Cloud, Kubernetes, OpenStack, OVH, etc.)
- Able to create own local private provider
- Simple command to create, update, and destroy the virtual clusters
  - `terraform apply` (create and update the cluster)
  - `terraform destroy` (destroy the cluster)
- Configuration is done in a real programming language (HashiCorp Language)
  - Not just YAML or JSON files
  - Enables proper Software Engineering practices – avoid repeating oneself
  - Enables custom modifications of the deployment procedure
  - Magic Castle has only one interface file to deploy clusters (`main.tf`)

Cloud-init

The “battle” between mutable vs immutable images

- Cloud-init allows the customization of images when they first boot
- Decreases the overhead of maintaining multiple custom images
  - Update the system packages
  - Create certificates
  - Install system packages – puppet in the case of Magic Castle
  - Create/modify/remove admin user
  - Configure initial security policies
  - No control over the final image package version and final application performances
- Used to reboot VMs just after initial config and update

Puppet

Infrastructure management tool

- Used by Compute Canada to maintain their own clusters
- Achieves the self-healing aspect of having the agent running on the nodes (self-healing)
- Individual node customizations are problematic or require large configuration changes
- Works great with the asynchronous provision requirement they have

HashiCorp Consul

Automate routine tasks

- Consul is used to maintain dynamic configurations
  - Slurm nodes are properly configured
  - CVMFS repositories are mounted appropriately
  - Configure Prometheus to monitor nodes
  - Configure nodes to use rsyslog
  - Configure nodes to use squid server
- Used by Magic Castle to maintain configuration upon addition/removal of nodes in the cluster

My experiences

My own experiences

Using Magic Castle

- Magic Castle worked out-of-the-box 100% of the time
  - Implement external courses about ReFrame
  - Deploy ephemeral cluster for application testing
  - Deploy a long-standing cluster for CI/CD
- Takes around 30 min to spawn a cluster
- Four different languages for configuring/customizing the cluster has its costs, but it paid all the dividends
  - Infrastructure definition (Terraform Cloud) separated from cluster config (GitHub project)

My own experiences

Using Magic Castle

- Terraform is great if you like coding
  - It allows a lot of code reuse
  - The issue is that every provider has its own set of APIs
- Cloud-init is extremely simple, no “big-deal” to learn it
- Puppet is great to self-heal the cluster
  - If you need it for this purpose, then there is something wrong
  - Can be completely replaced by other tools, like Ansible, Salt, Chef, etc.
- Consul is the “magical” tool
  - Keeps Slurm config up to date – acts upon node removals and additions to the cluster
  - The same for the DNS and Prometheus
  - It is a great synergy when used with “immutable” images

My own experiences

Using Magic Castle

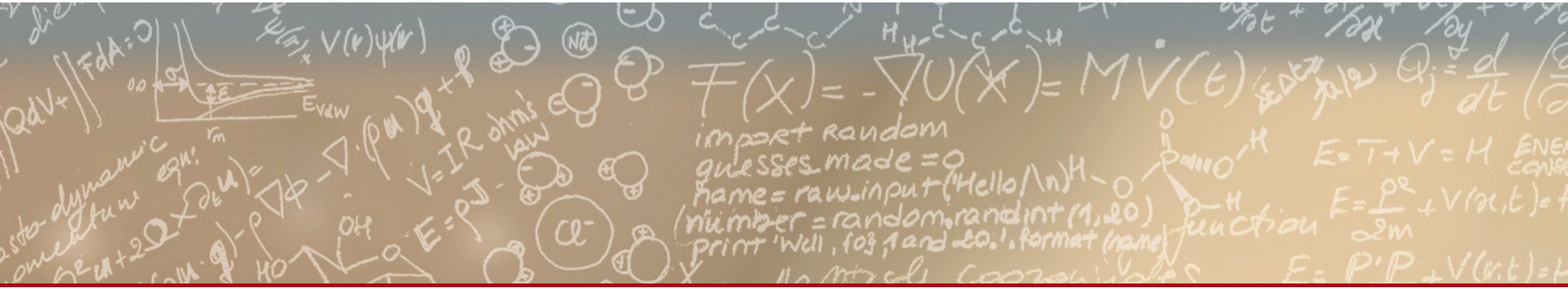
- Using immutable images
  - is a question of not using Puppet and Cloud-init
  - “just” modify Terraform
  - Consul can still be used to maintain the cluster 100% self-maintained
- Updating the system constantly
  - Doesn’t affect the performance of the scientific software provided by the Compute Canada or EESSI stack
  - But it does affect if one compiles the code locally
- Turning SELinux on and off change application performance by less than 2%
  - Tested about 50 different applications provided by Compute Canada
- FreeIPA fails too often when the system gets too much load
  - Changed to use /etc/hosts and local accounts

Conclusion

Conclusion

Use Magic Castle

- Do not be afraid of using multiple configuration management tools
- Use the right tool for the right job
- Terraform is great for infrastructure management
- Cloud-init is great for modifying images when booting for the first time
- Puppet is great for system configuration, but there are other alternatives
- Consul is great for maintain the system configured
- Magic Castle is a great tool for teaching and experimenting with HPC
- Magic Castle is good for tuning up sys admin skills, if you want to ☺



Thank you for your attention.

Can we isolate the root causes of the performance changes?

The PoC workflow

- Write ReFrame checks that invokes another ReFrame instance and instantiates one Magic Castle cluster per test
- Run reframe checks in each individual cluster
- ReFrame decommissions each individual cluster and collects all test data