



# Managing the CERN Batch System with Kubernetes

**Speaker:** Luis Fernandez Alvarez,  
Ben Jones, Olga Datskova and Gavin McCance

24<sup>th</sup> International Conference on Computing in  
High Energy & Nuclear Physics

# Batch Service *restyling*

# The Batch Service @ CERN IT

*Provides Tier-0 compute power via HTCondor to WLCG.*

- Process CPU intensive workload ensuring fairshare among various user groups
- Maximize utilization, throughput, efficiency
- It runs jobs from the Grid and from local CERN departments

**235K**

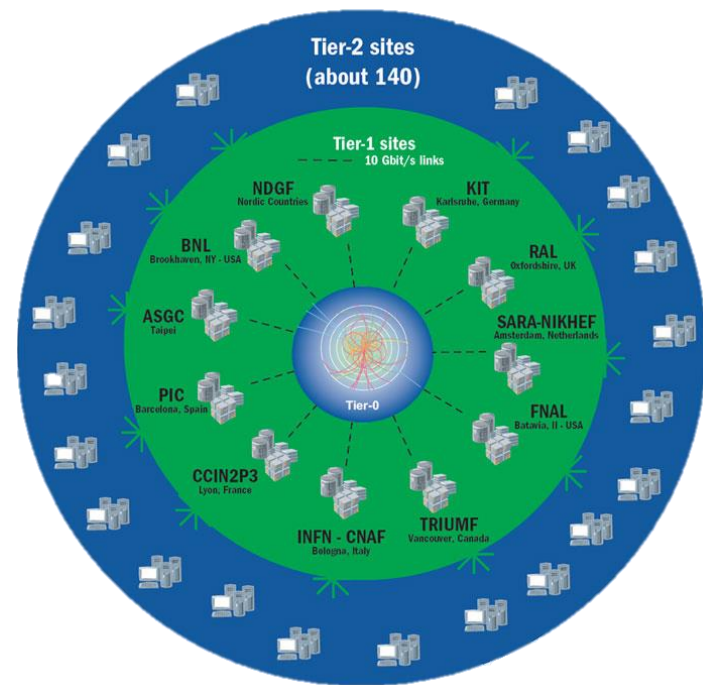
Total  
Cores

**20K**

Virtual  
Machines

**1M**

Completed /  
Day



# From 2012 to 2019

## Context

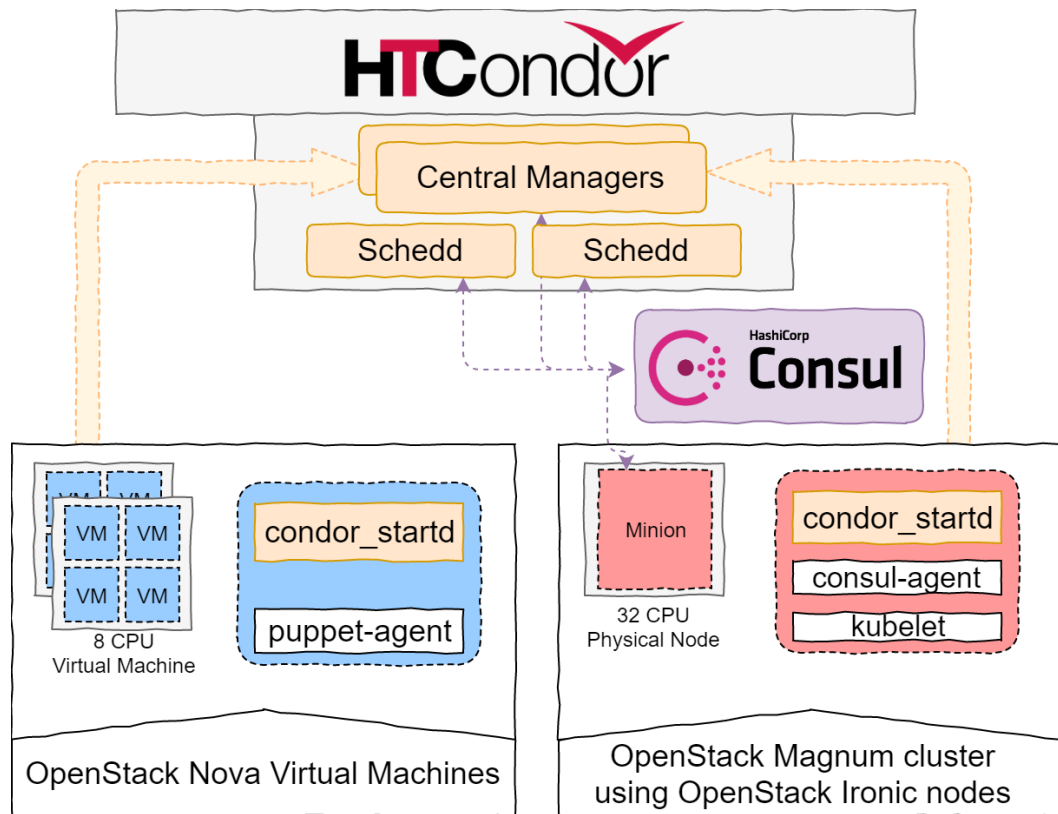
- From Cloud APIs spawning VMs to baremetal provisioning in the Cloud
- Not just one private cloud anymore (HNSciCloud, OCRE...)
- Virtual machines vs containers
- From Puppet to a wide ecosystem of configuration tools around Kubernetes
  
- Compute demands growing
- Budget still tight

## Motivation

- High compute demands and low budget, we must optimize what we have...  
*run on baremetal profiting from the Cloud APIs?*
  
- Many opportunistic resource scenarios, we need to be flexible
- Reduce time spent doing operations...  
*make compute provisioning less couple to CERN via Kubernetes?*

# Prototype

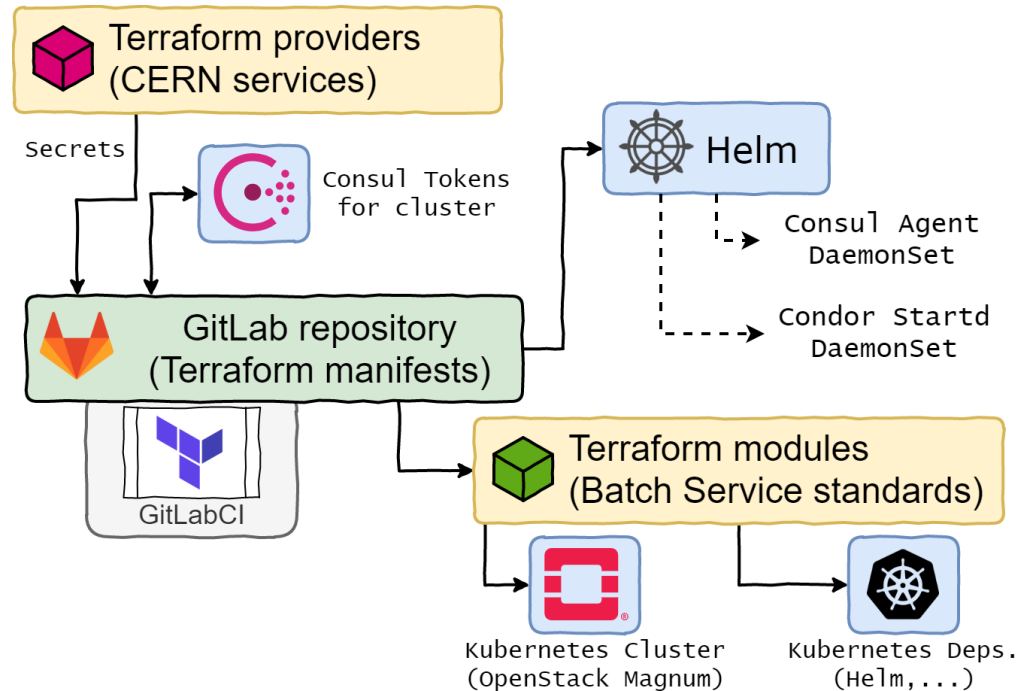
- Hybrid HTCondor cluster with worker nodes in two forms:
  - Traditional Puppet managed VMs
  - Kubernetes based minions
- New element: Consul.
- Total capacity of 100 x 32CPU (SMT-ON) machines:
  - 50%: Puppet 8 CPU virtual machines
  - 50%: Kubernetes baremetal nodes
- Many areas of interest to evaluate:
  - **Operations impact**
  - **Resource usage efficiency**



# Service operations

# Ops: Bootstrapping

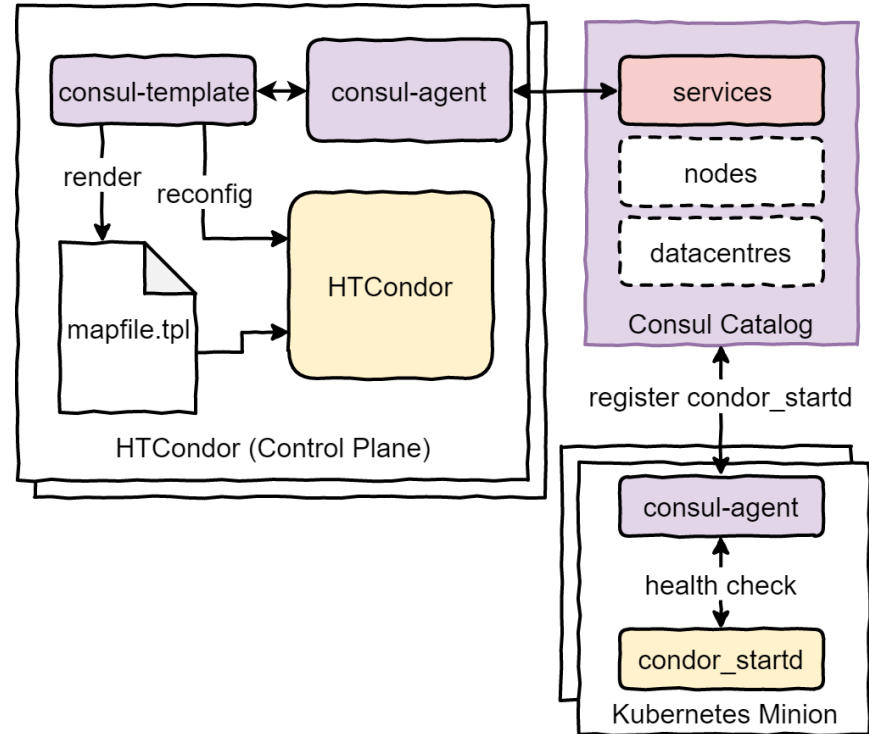
- Before: homegrown scripts and Puppet
- Terraform based deployment.
  - Extended via custom modules and providers
- Infrastructure vs Application
- Automation via GitLabCI





# Ops: Discovery & Authentication

- The previous PuppetDB based discovery of worker nodes is replaced by....
- ...Consul based discovery
  - `condor-startd` service defined
  - Control plane using `consul-template` to populate configuration
  - Mixed PuppetDB & consul-template for backwards compatibility
- Daemon authentication:
  - GSI in both modes:
    - OpenStack Magnum customized to provide Grid certificates in minions
  - TOKEN auth evaluated in k8s mode



# Other operations

- **Draining (node lifecycle)**, exploring a combination of:
  - Moving from homegrown tooling (roger) to Kubernetes built-in node labeling functionality
  - Use of other tools like node-problem-detector to modify minion states
  - Benefit from Consul KV and tools like envconsul to populate Condor state
- **Upgrades**
  - Still not yet decided the right approach for workers upgrade
  - Move from fat workers to lightweight instances via HTCondor transforms to wrap jobs within Singularity/Docker images
  - Reduce the impact of future migrations (CentOS8) on cluster availability

# Resource usage efficiency

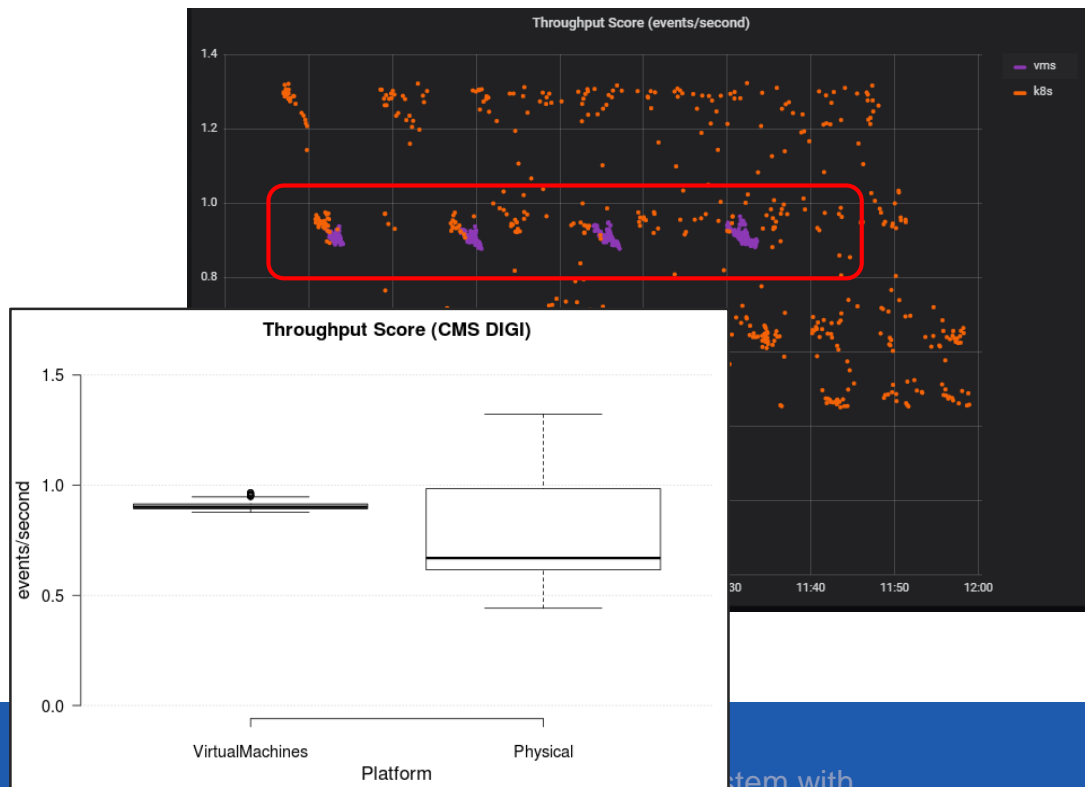
# Benchmarking

*!! See CHEP2019 talk: [Using HEP experiment workflows for the benchmarking and accounting of computing resources](#)*

- Evaluate performance of both models
- Benefit from the current effort of the Benchmark WG: [hep-workloads](#).
- Benchmarks submitted as HTCondor jobs:
  - 1600 cores per platform, CentOS7 workers.
  - 8 core jobs. Benchmark payload depending on the benchmark:
    - Single-threaded: 1 thread x 8 copies
    - Multi-threaded: 8 threads x 1 copy
  - 800 jobs per platform (VMs vs Kubernetes): resources filled 4 consecutive times
  - Mainly executed as Singularity jobs (SLC6 based benchmarks)
- Results sent to the CERN IT monitoring infrastructure to be indexed in ElasticSearch and visible via Grafana

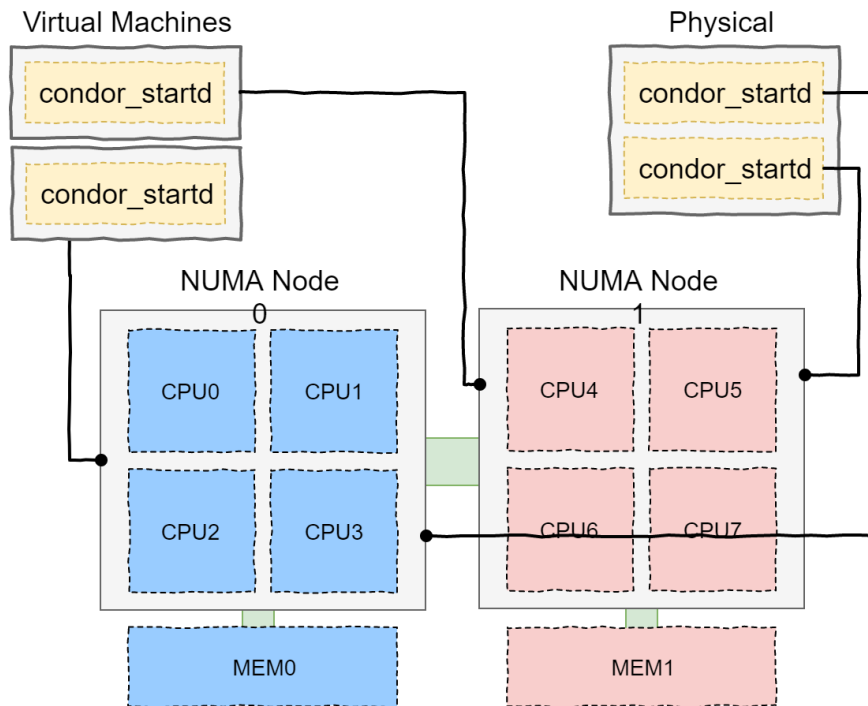
# Unexpected results

- Very spread results on baremetal.
- Total throughput lower in some benchmarks:
  - 800 Jobs.
  - VMs 1.5 times faster.
- Configuration review...



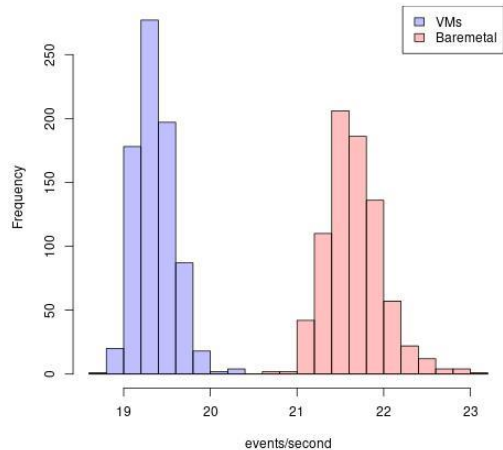
# NUMA awareness

- We have to help the kernel scheduling by pinning processes to NUMA nodes.
  - Already solved in the CERN cloud by scheduling VMs to NUMA nodes: Optimisations of the Compute Resources in the CERN Cloud Service.
- **condor\_startd on VMs:** automatically tied to NUMA node as VM already is
- Apply same principle to **condor\_startd on physical nodes:** instantiate one daemon per NUMA node
- Use cpusets to confine each daemon
- Exposed via HTCondor as multiple slots:  
`slot1@numa0@<hostname>`  
`slot2@numa1@<hostname>`

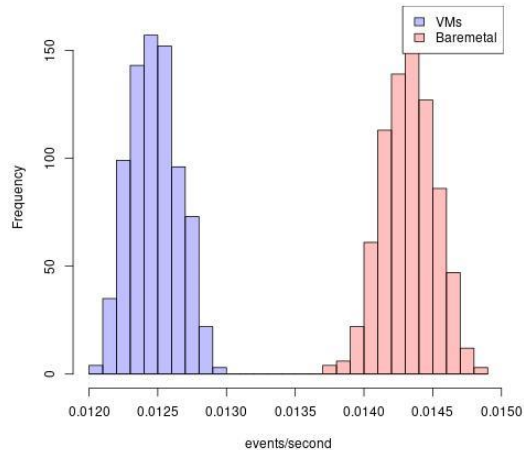


Difference between benchmarking in Singularity image vs accessing CVMFS? to be explored...

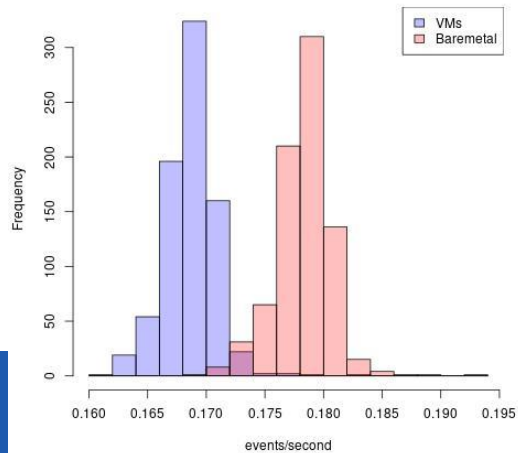
Throughput Score (LHCb-GEN-SIM)



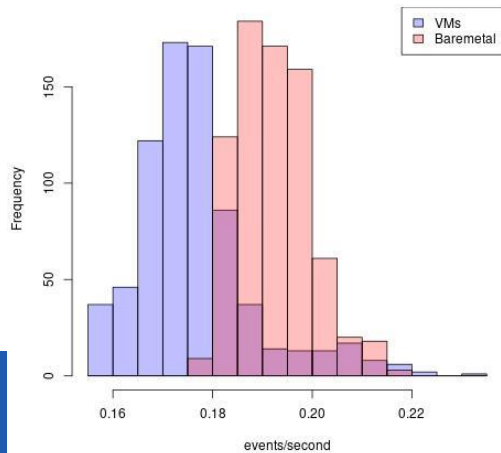
Throughput Score (ATLAS-SIM)



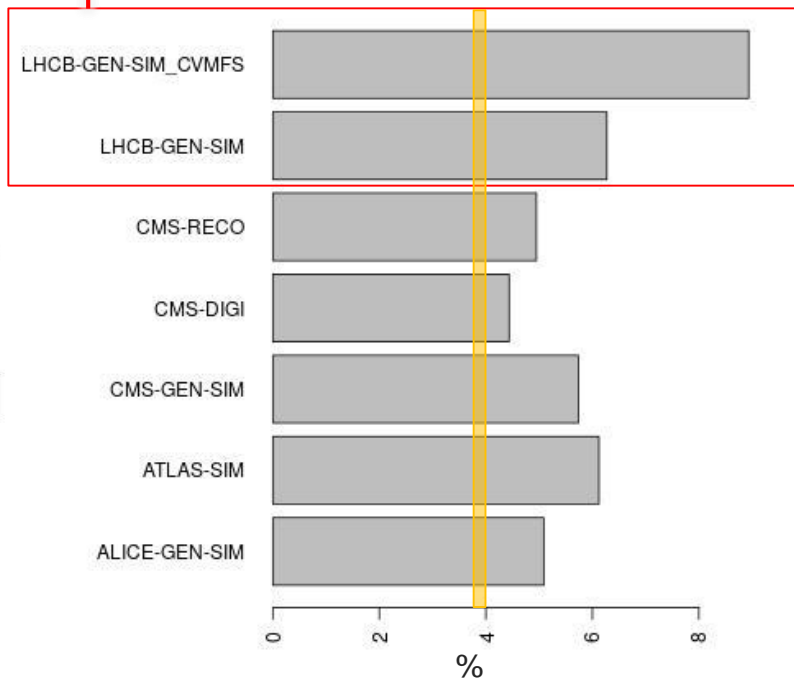
Throughput Score (CMS-GEN-SIM)



Throughput Score (ALICE-GEN-SIM)



Benchmark throughput improvement on baremetal



# Conclusions & next steps

- New model has proved to be operationally feasible
  - Benefiting from Kubernetes built-in functionality can make our service more efficient but...
  - Switching not trivial: it would have an impact on existing workflows in other IT services (monitoring, datacentre operations,...)
- Moving to baremetal infrastructure could give us 5% more capacity in the existing resources

Iterate and improve the proposed prototype:

- Deploy a similar setup in our production HTCondor cluster running production workloads.
- Extend the adoption of k8s to HTCondor control plane.
- Explore alternatives to expose the k8s clusters to HTCondor.
- Design how to integrate the benchmarks as a regular task in our service.



# Acknowledgement

## CERN Cloud Team

*Thanks for the help and work in the Magnum/K8s deployments, as well as providing the required resources via Ironic for this prototype.*

## Benchmarking WG1

*Thanks for the job on providing the different experiment workloads, it helped getting good insights out of this work.*

## CERN Batch Team

*This work is the result of the team effort improving our automation, containerization and operation procedures.*



Thank you