
CERN

Managing the CERN Batch System with Kubernetes

Luis Fernandez Alvarez / *pre-GDB – kubernetes many faces*

CHEP 2019: Managing the CERN Batch System with Kubernetes

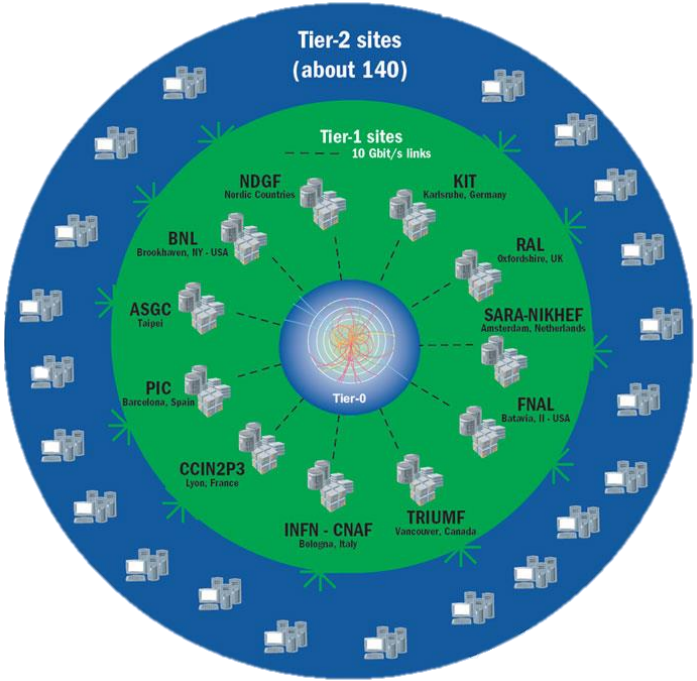
Batch Service *restyling*

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	20K	1M
Total Cores	Virtual Machines	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…

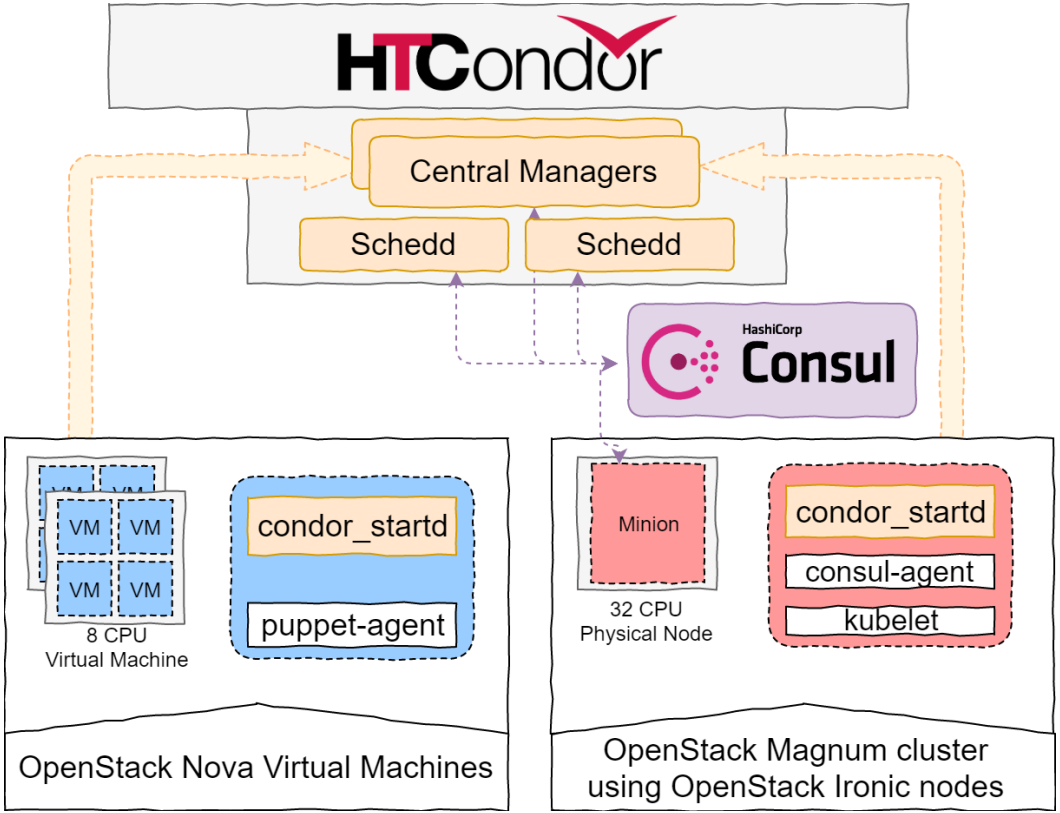
run on baremetal profiting from the Cloud APIs?

- Many opportunistic resource scenarios, we need to be flexible
- Reduce time spent doing operations…

make compute provisioning less couple to CERN via Kubernetes?

Prototype

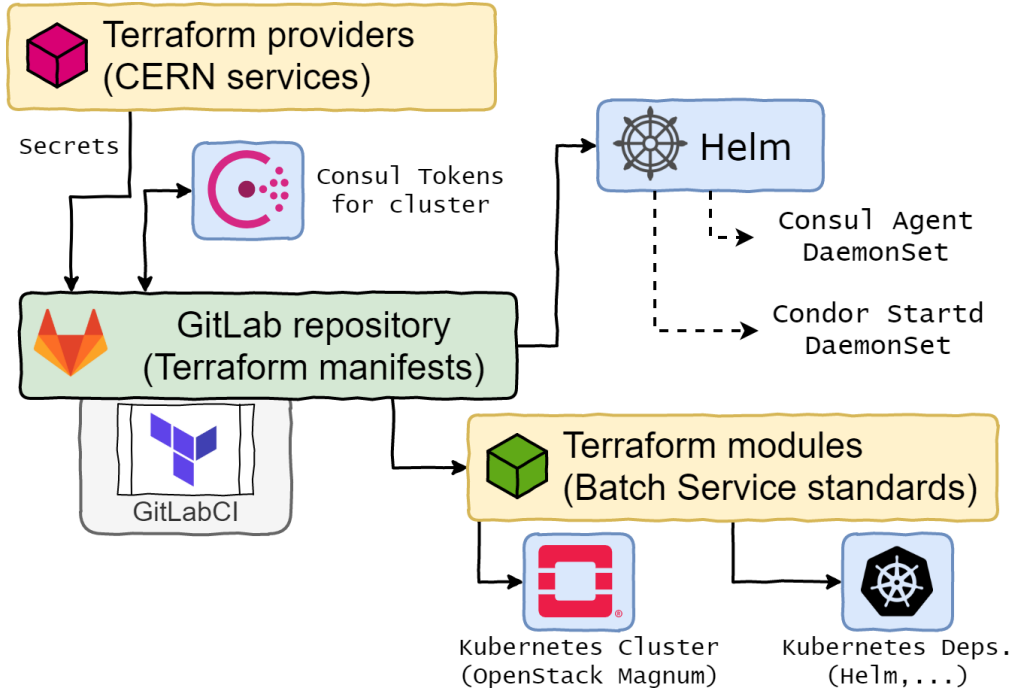
- Hybrid HTCondor cluster with worker nodes in two forms:
  - Traditional Puppet manged VMs
  - Kubernetes based minions
- New element: Consul.
- Total capacity of 100 x 32CPU (SMT-ON) machines:
  - 50%: Puppet 8 CPU virtual machines
  - 50%: Kubernetes baremetal nodes
- Many areas of interest to evaluate: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



Service operations

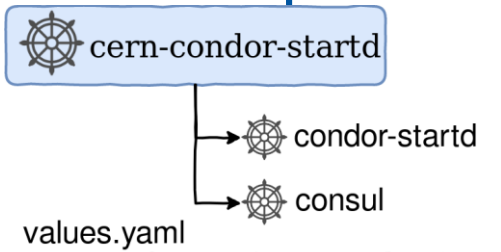
Ops: Bootstrapping

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



Ops: Bootstrapping (II)

- Helm composition



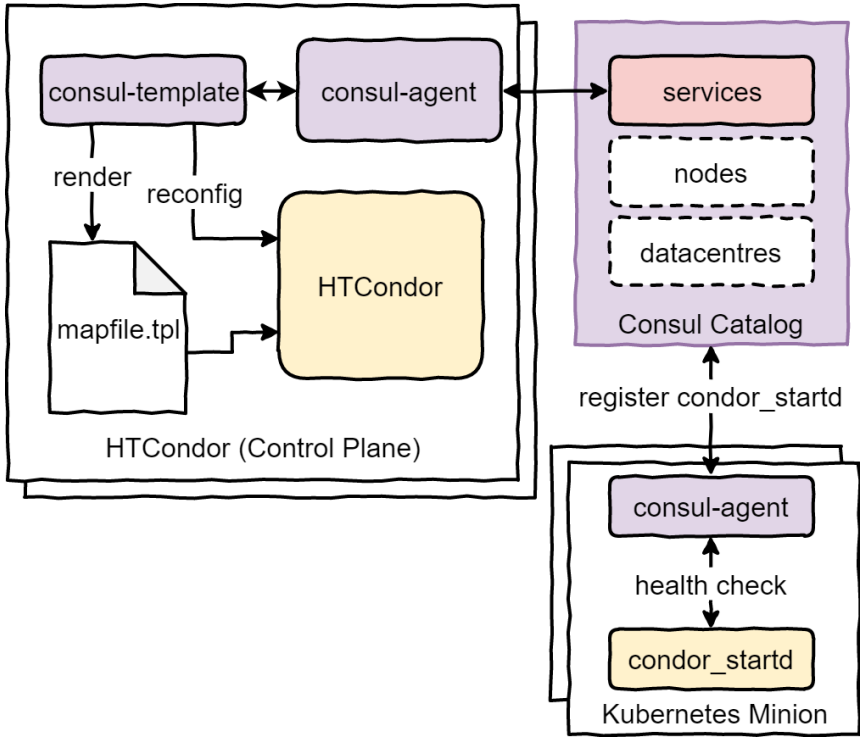
values.yaml

```
---
zone: "IT-Batch - PT8 Project 006 - Physical"
condor-startd:
  infra:
    managers:
      - tweetybird03.cern.ch
consul:
  client:
    image: 'gitlab.cern.ch...'
```

- To be explored:
  - Multi-Cluster
  - Cluster autoscaling in opportunistic resources?
  - Different Helm CI/CD workflows: Flux?

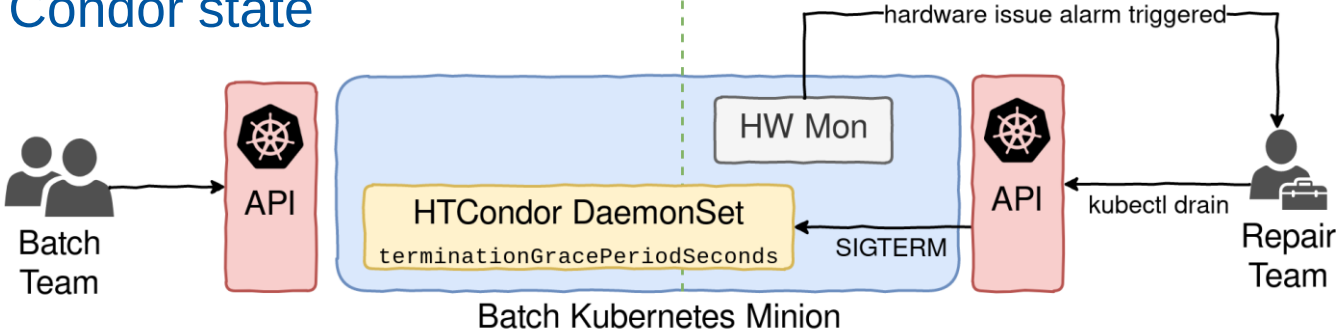
Ops: Discovery & Authentication

- 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 (I)

- Draining (node lifecycle), exploring a combination of:
  - Moving from homegrown tooling (roger) to Kubernetes built-in node draining and labeling functionality
  - Use of other tools like node-problem-detector to modify minion states (drain automatically: draino)
  - Benefit from Consul KV and tools like envconsul to populate Condor state



Other operations (II)

- 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

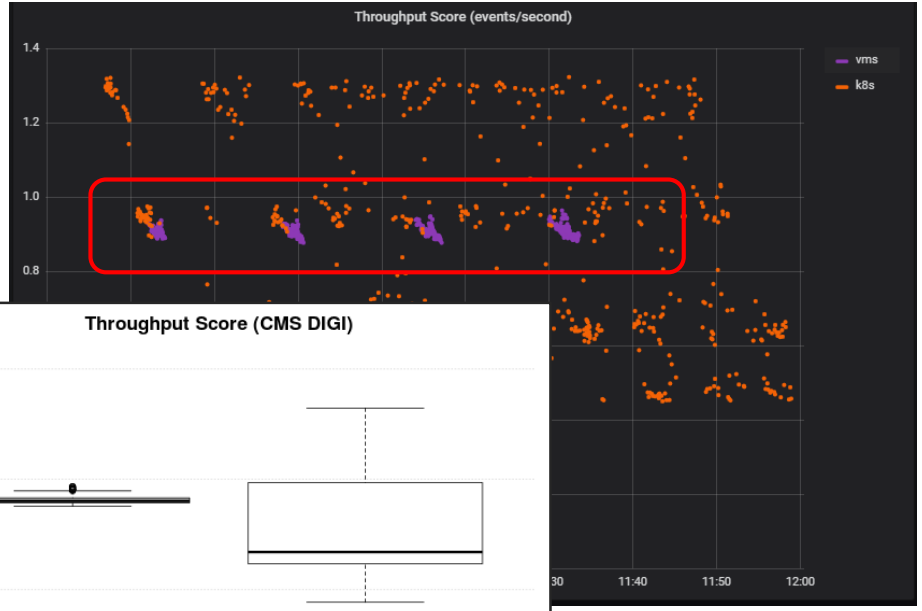
Benchmarking

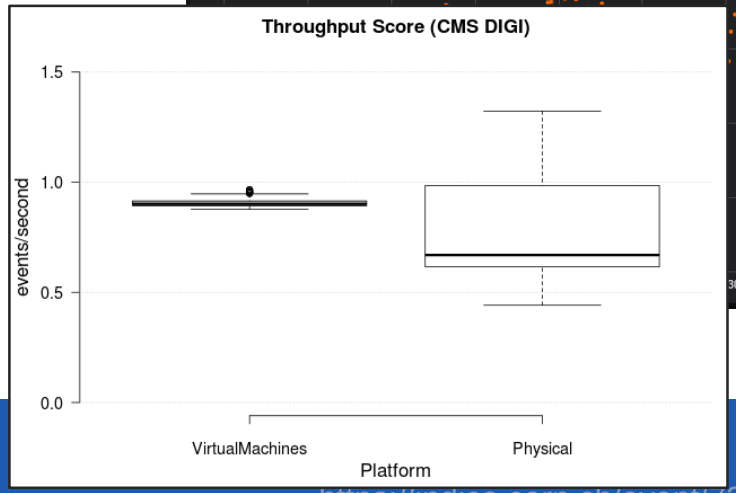
!! See CHEP2019 talk Using HEP experiment workflows for the benchmarking and accounting of computing resources

- 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

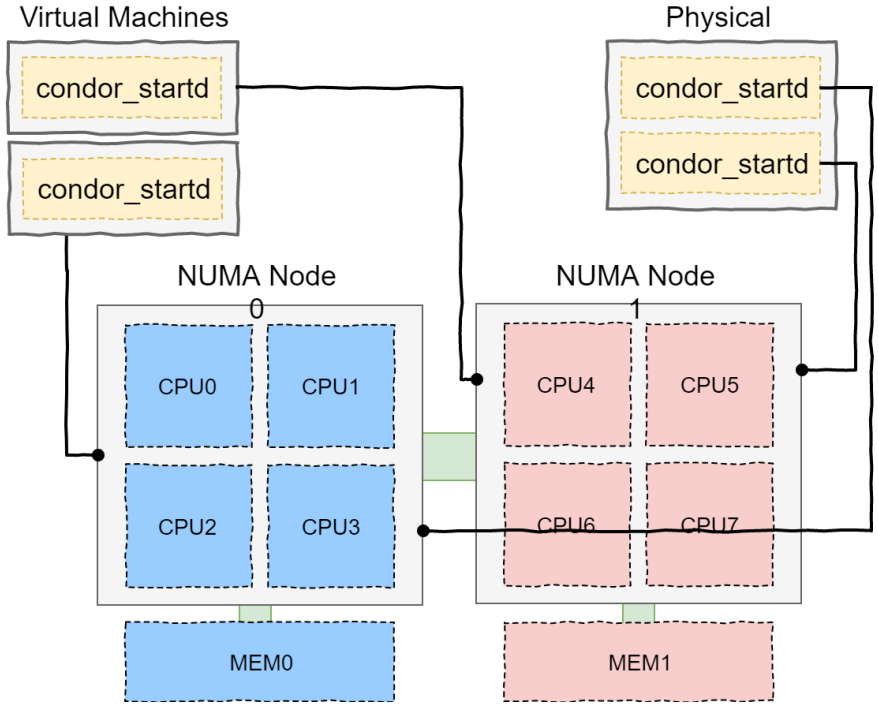
- Very spread results on baremetal.
- Total throughput lower in some benchmarks:
  - 800 Jobs.
  - VMs 1.5 times faster.
- Configuration review…

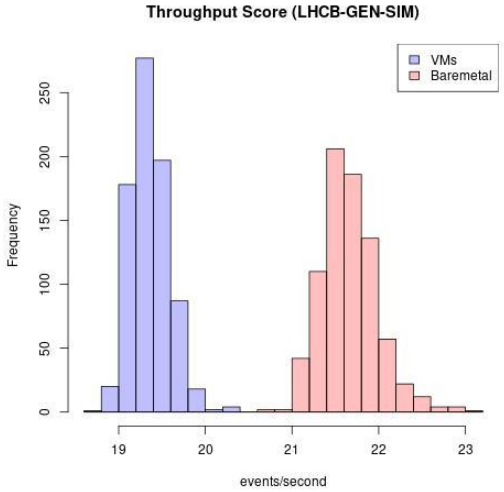


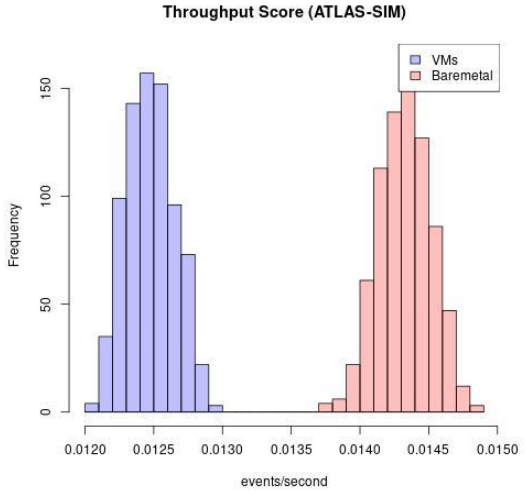


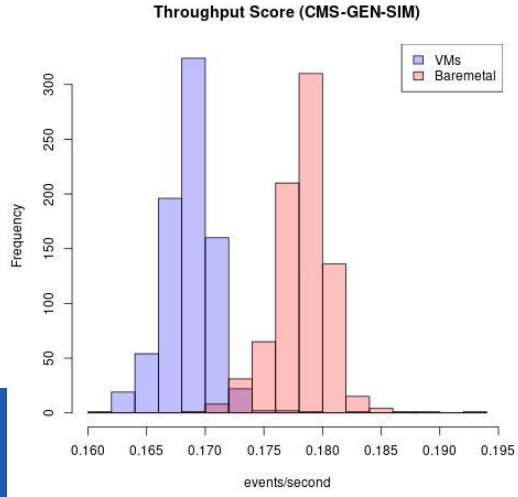
NUMA awareness

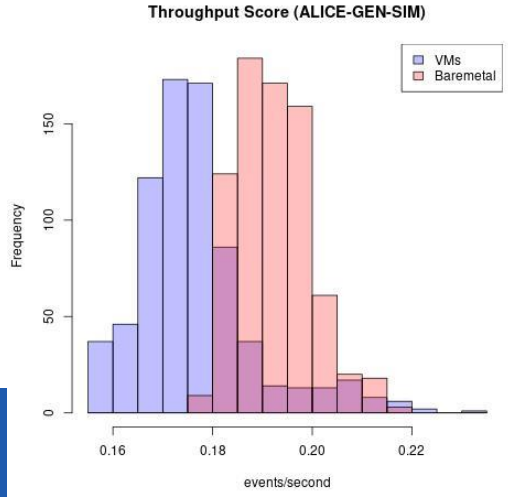
- We have to help the kernel scheduling by pinning processes to NUMA nodes.
  - Already solved in the CERN cloud by scheduling VMs to NUMA nodes: Optimisations of the Compute Resources in the CERN Cloud Service.
- `condor_startd` on VMs: automatically tied to NUMA node as VM already is
- Apply same principle 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>`



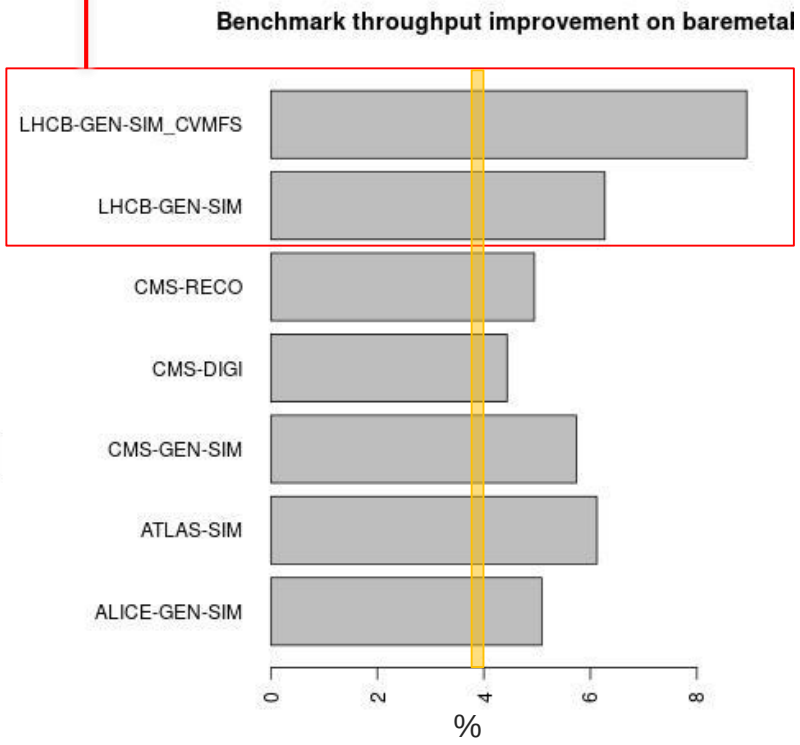








Difference between benchmarking in Singularity image vs accessing CVMFS? to be explored…



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, datecentre 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: Helm, operators…
- Explore alternatives to expose the k8s clusters to HTCondor: condor blahp.
- Design how to integrate the benchmarks as a regular task in our service.

Thank you