KubeCon | CloudNativeCon

Europe 2021

Virtual

Forward Together »

KubeCon | CloudNativeCon
Europe 2021
Virtual

Building and Managing a Centralized ML Platform with Kubeflow at CERN

Dejan Golubovic, Ricardo Rocha

Who

Dejan Golubovic - dejan.golubovic@cern.ch

Computing Engineer in the CERN Cloud team

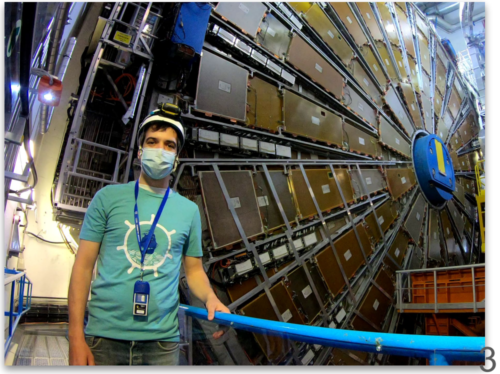
Focus on Machine Learning

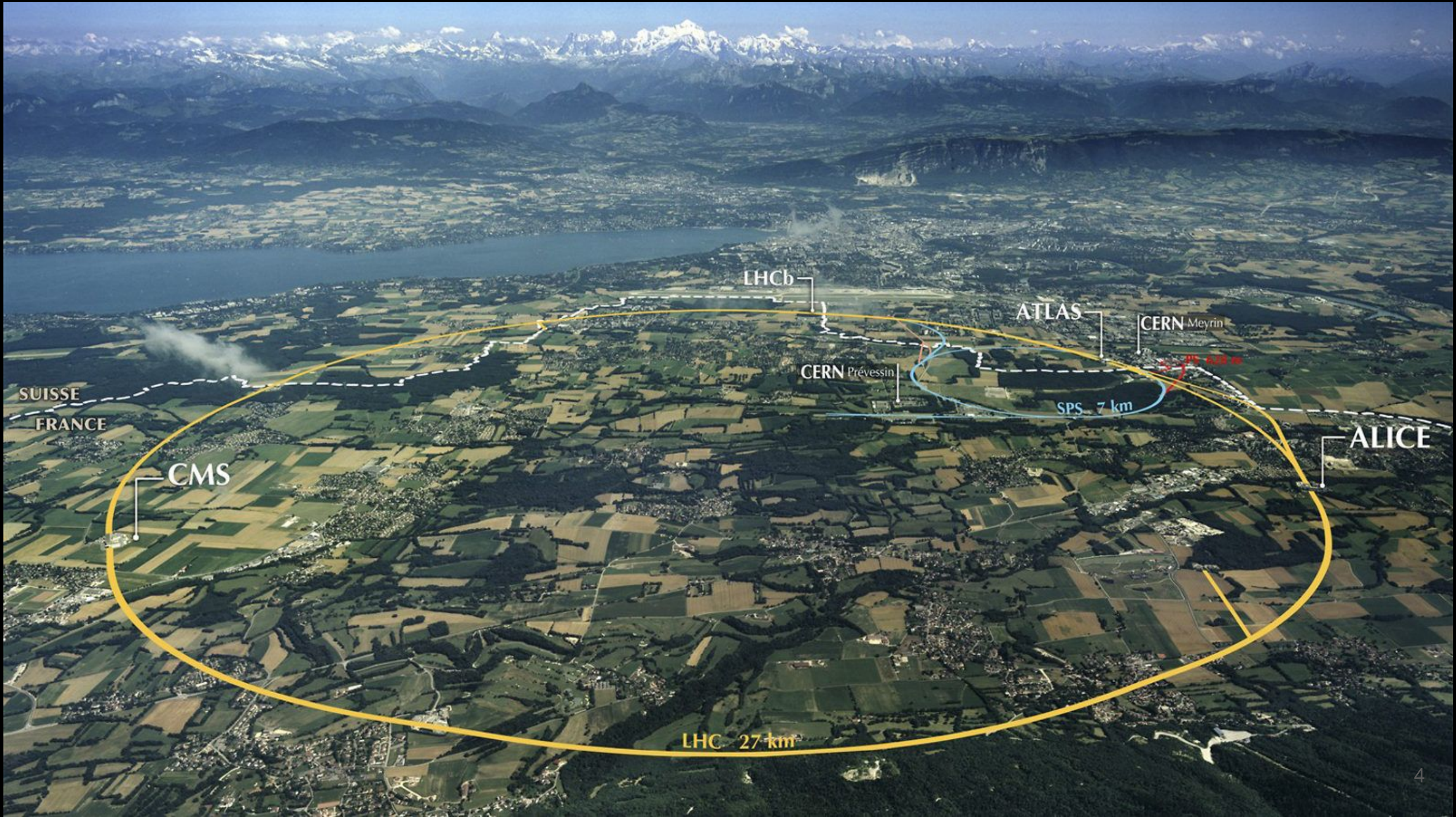
Ricardo Rocha - ricardo.rocha@cern.ch , @ahcorporto

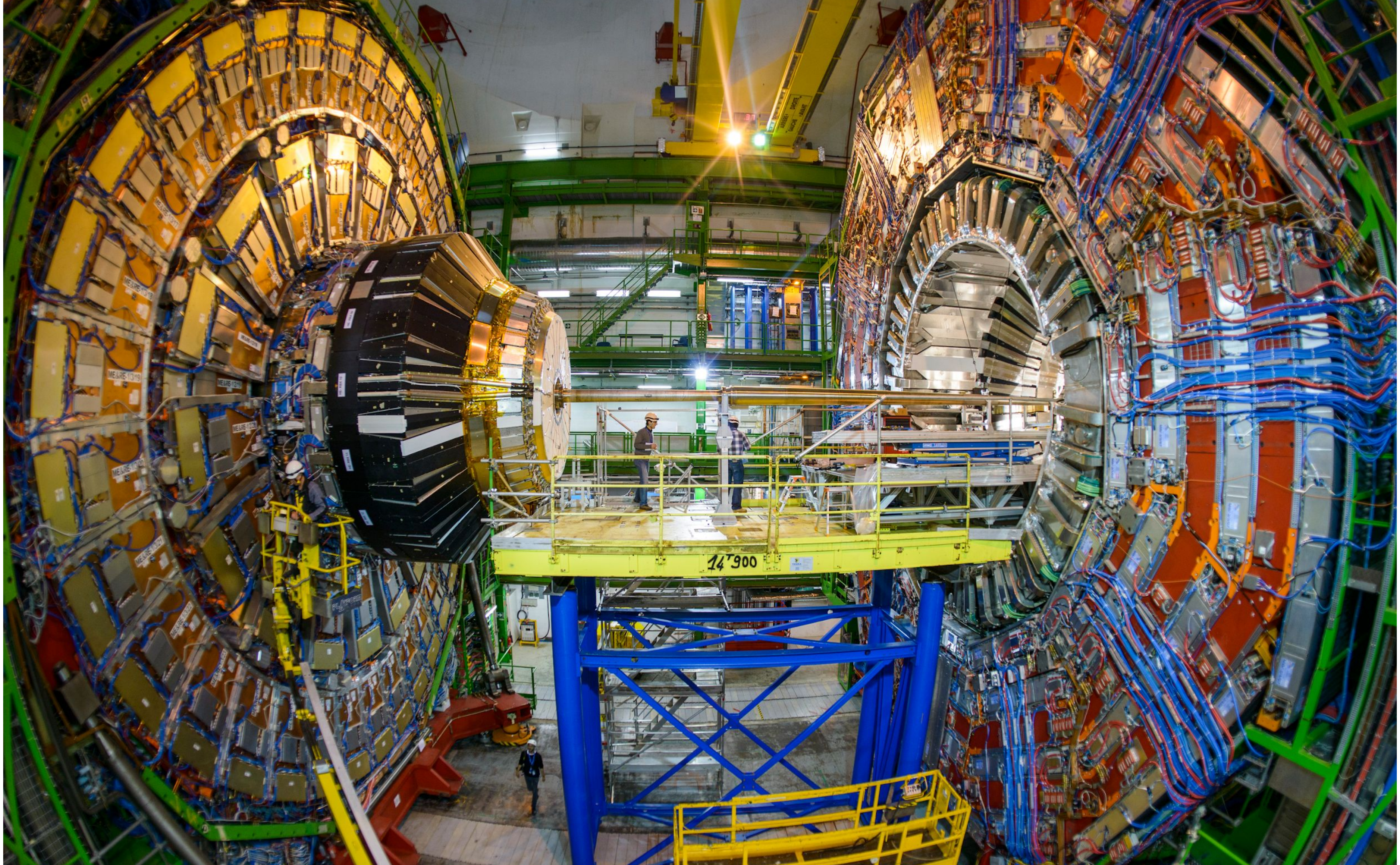
Computing Engineer in the CERN Cloud team

Containers, networking, GPUs/accelerators and ML

CNCF TOC




LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
LHC 27 km


14'900

Motivation

Machine Learning is taking a big role in High Energy Physics

Resources like GPUs are currently too spread, and so is knowledge

Physicists are not (necessarily) infrastructure experts

Use Cases

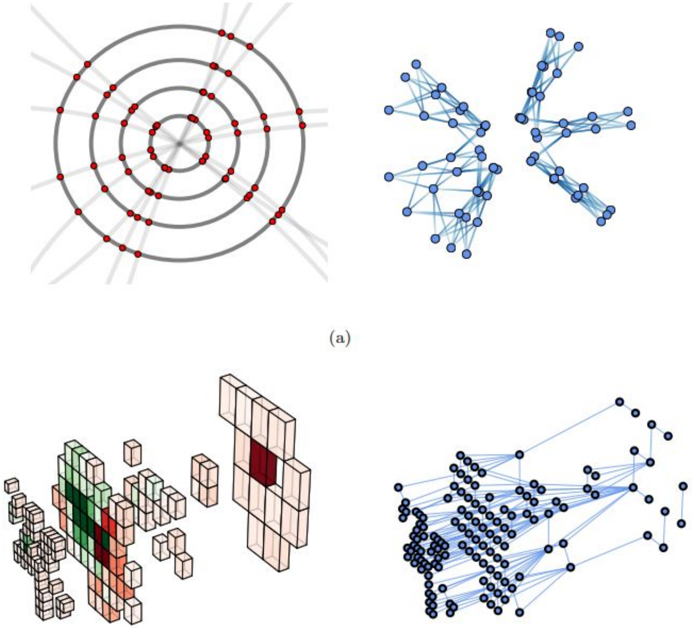
Particle Tracking / Reconstruction

Graph Neural Networks (GNNs) for

event reconstruction

Track finding and fitting in the detectors

https://arxiv.org/pdf/2012.01249.pdf



(a)

(b)

Fig. 3. HEP data lend themselves to graph representations for many applications: segments of hits in a tracking detector hits (a), and neighboring energy deposits in calorimeter cells (b). Figures reproduced from Ref. [41].

Use Cases

Fast Simulation with 3D GANs

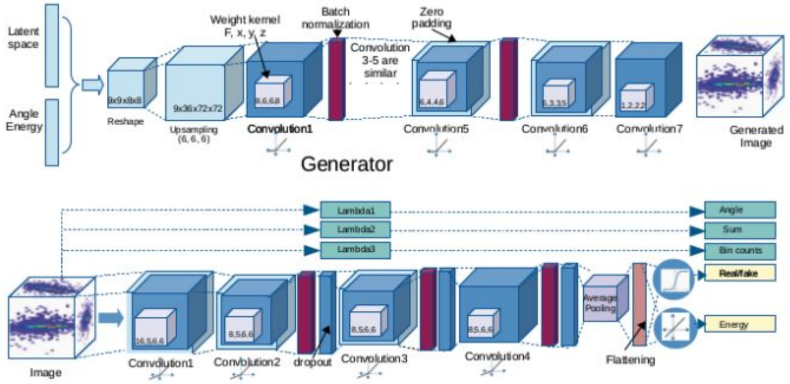
Tackle the upcoming challenges of High Luminosity LHC

10x more data coming soon

Alternative to traditional Monte Carlo

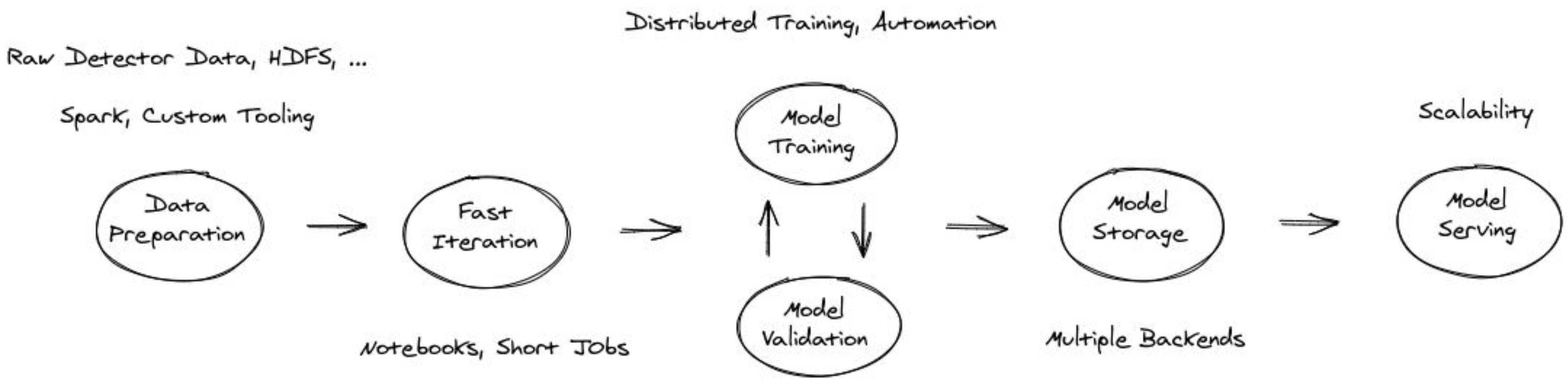
No need to store data, simulate it on the fly

20000x speed up

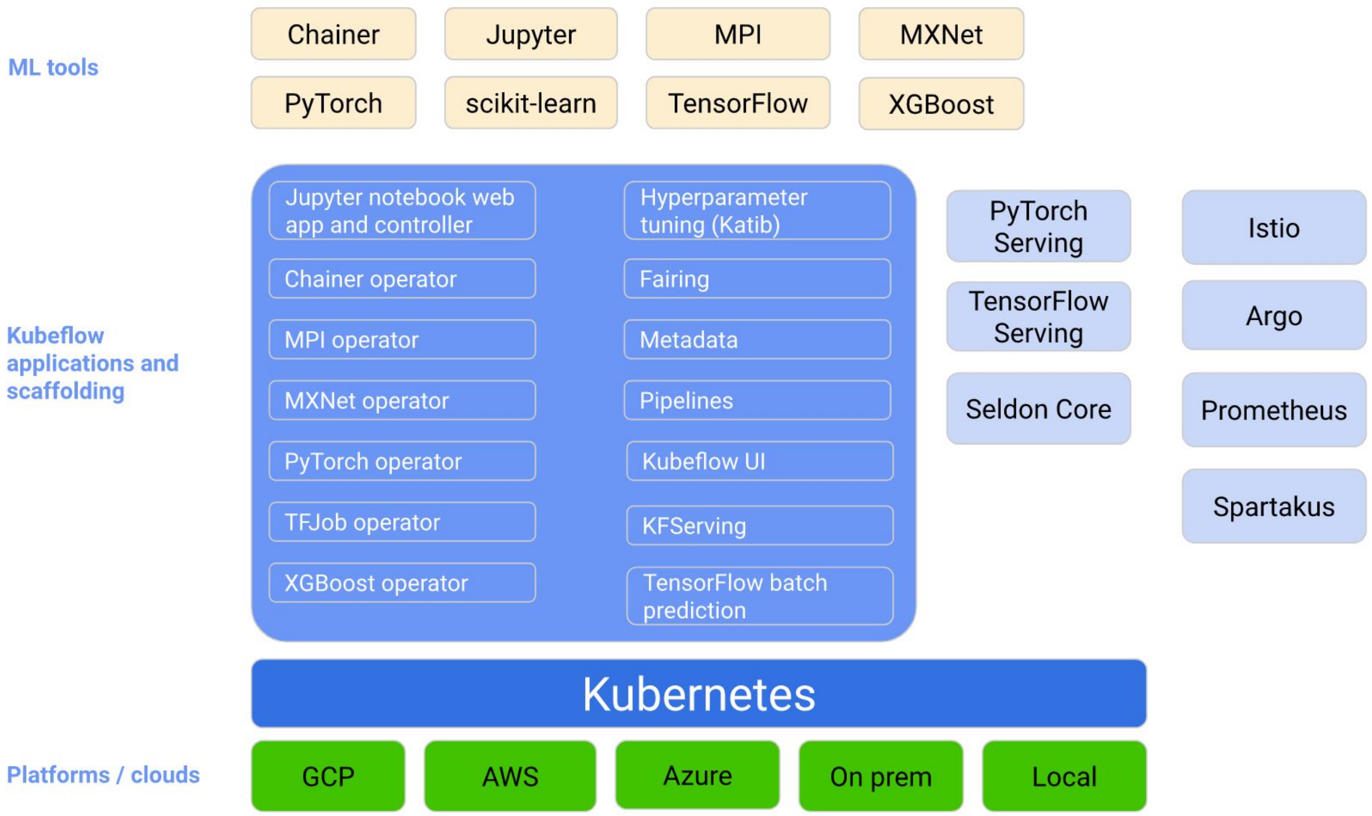


https://iopscience.iop.org/article/10.1088/1742-6596/1525/1/012064/pdf

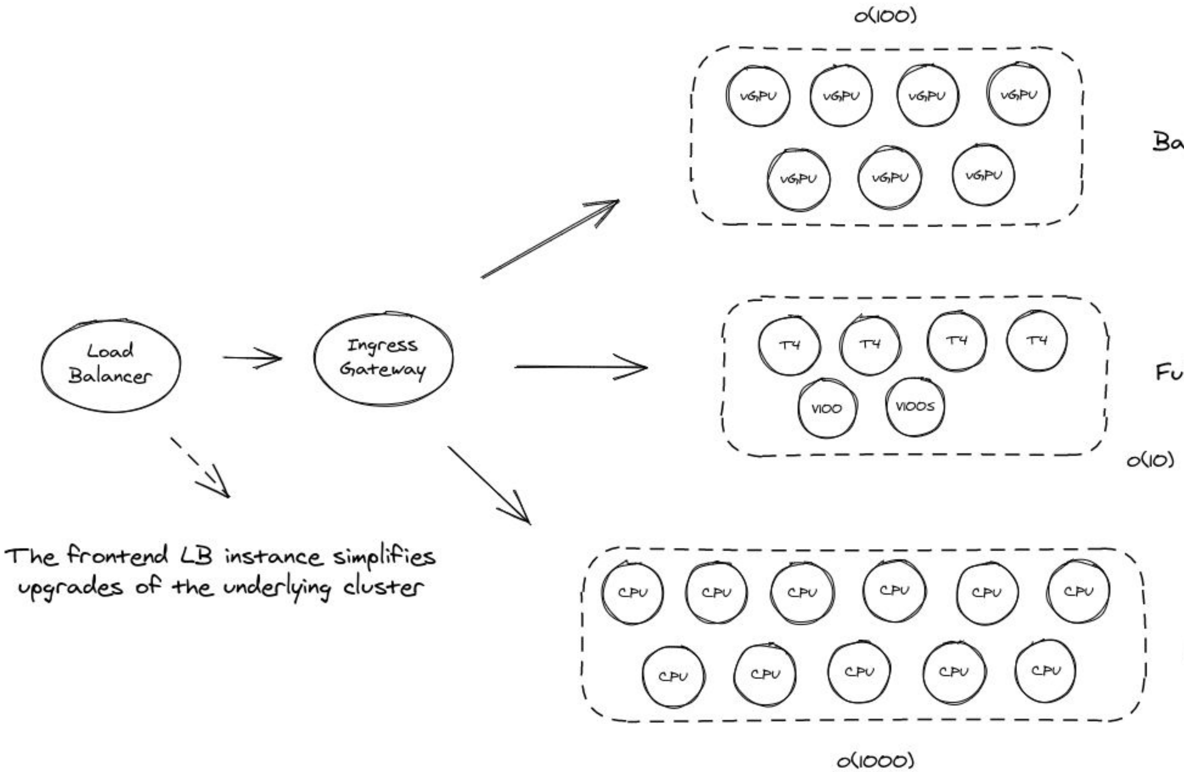
Goal : Platform to manage the full machine learning lifecycle



Kubeflow


ML tools
Chainer
Jupyter
MPI
MXNet
PyTorch
scikit-learn
TensorFlow
XGBoost
Kubeflow applications and scaffolding
Jupyter notebook web app and controller
Hyperparameter tuning (Katib)
Chainer operator
Fairing
MPI operator
Metadata
MXNet operator
Pipelines
PyTorch operator
Kubeflow UI
TFJob operator
KFServing
XGBoost operator
TensorFlow batch prediction
PyTorch Serving
TensorFlow Serving
Seldon Core
Istio
Argo
Prometheus
Spartakus
Kubernetes
Platforms / clouds
GCP
AWS
Azure
On prem
Local

Cluster(s) Layout



vGPUs are used mostly for notebooks and quick iteration

Based on T4s, with time sharing of 4 virtual instances per card

Exposed to users via PCI passthrough
Full access used mostly for pipelines, hyper parameter optimization and model serving
No NVLink / fast interconnects

Workloads other than deep learning can also benefit from the platform - notebooks / jupyter envs, generic pipelines, recurrent pipeline jobs...

Deployment

Kubernetes 1.18, Kubeflow 1.1, Istio 1.5, Knative 0.15.0

GitOps with ArgoCD managing multiple applications per environment

- Kubeflow using kustomize
- Istio and Nvidia drivers / licenses with operators
- Prometheus, Knative, cert-manager using Helm charts

Integrations

Auth / Authz done using CERN SSO / OIDC, based on Keycloak

- Internal groups mapped to roles
- User ID and assigned roles mapped to Kubeflow profiles / namespaces
- Default quotas on personal namespaces (fixed), flexible for group profiles

A variety of **storage systems**

- CernVM FS, a read-only set of hierarchical caches for sw distribution
- EOS for physics data: both krb5 and OAuth2 based access available
- HDFS, mostly used for data preparation with Spark, krb5 based access

Issues

Releases not always consistent in terms of functionality

- Ex: 1.1 brought multi user pipelines, but broke other components (i.e. kale)
- Couple weeks to sort out downstream the different integrations

Kustomize based deployment hard to dig into

- Simplified things by removing some components from the bundle: cert-manager, istio, knative
- Allow for different versions from the bundled dependencies - this ended up as a requirement
- Only kubeflow apps managed by kustomize, overlays for prod and staging

Managing additional package requirements (both in notebooks and pipelines)

Bursting

Bursting out is key for our deployment

- (Much) Larger amounts of GPUs, specialized accelerators (TPUs, IPUs)

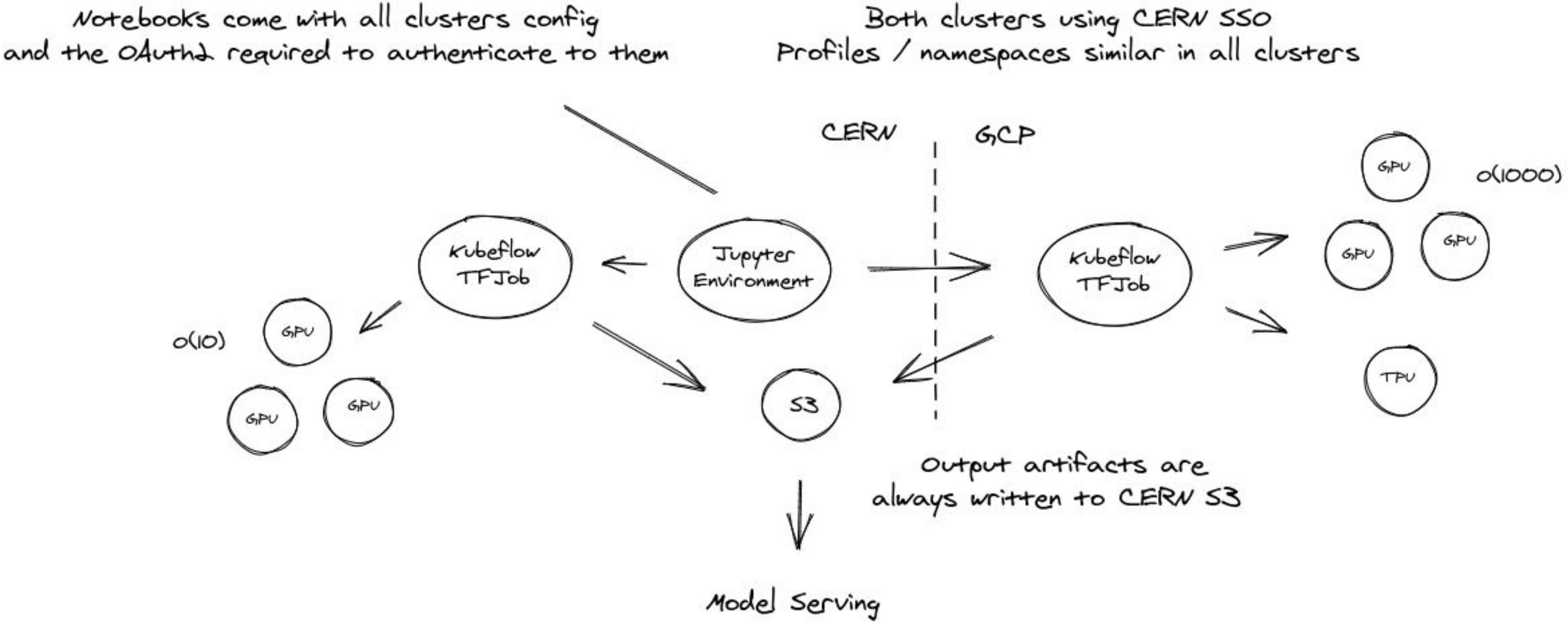
Several attempts to do it at a lower level

- Federation, Virtual Kubelet, Istio Gateways, … moderate success

Promising Results: Expose clusters from inside Kubeflow Jupyter environments

- Jupyter servers get the cluster configs via a volume mount
- Users can choose a cluster, auth/authz done using the same OAuth2 token
- OPA to validate which groups / roles can submit to different clusters

Bursting



Notebooks come with all clusters config
and the OAuth2 required to authenticate to them
Both clusters using CERN SSO
Profiles / namespaces similar in all clusters
CERN
GCP
Kubeflow TFJob
Jupyter Environment
Kubeflow TFJob
GPU
GPU
GPU
o(10)
GPU
GPU
GPU
o(1000)
TPU
S3
Output artifacts are
always written to CERN S3
Model Serving

Demo: 3DGAN Training

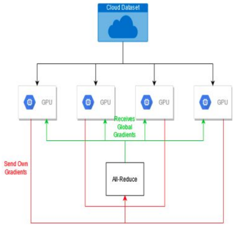
Extensive training time

- Full training of a single model: ~2.5 days

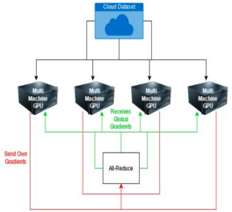
Solution - distributed training

- Use TensorFlow distributed Strategy tf.distribute.Strategy

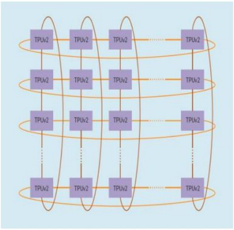
Mirrored Strategy



Multi Worker Mirrored Strategy



TPU Strategy



Demo: 3DGAN Training

Automate distributed training process

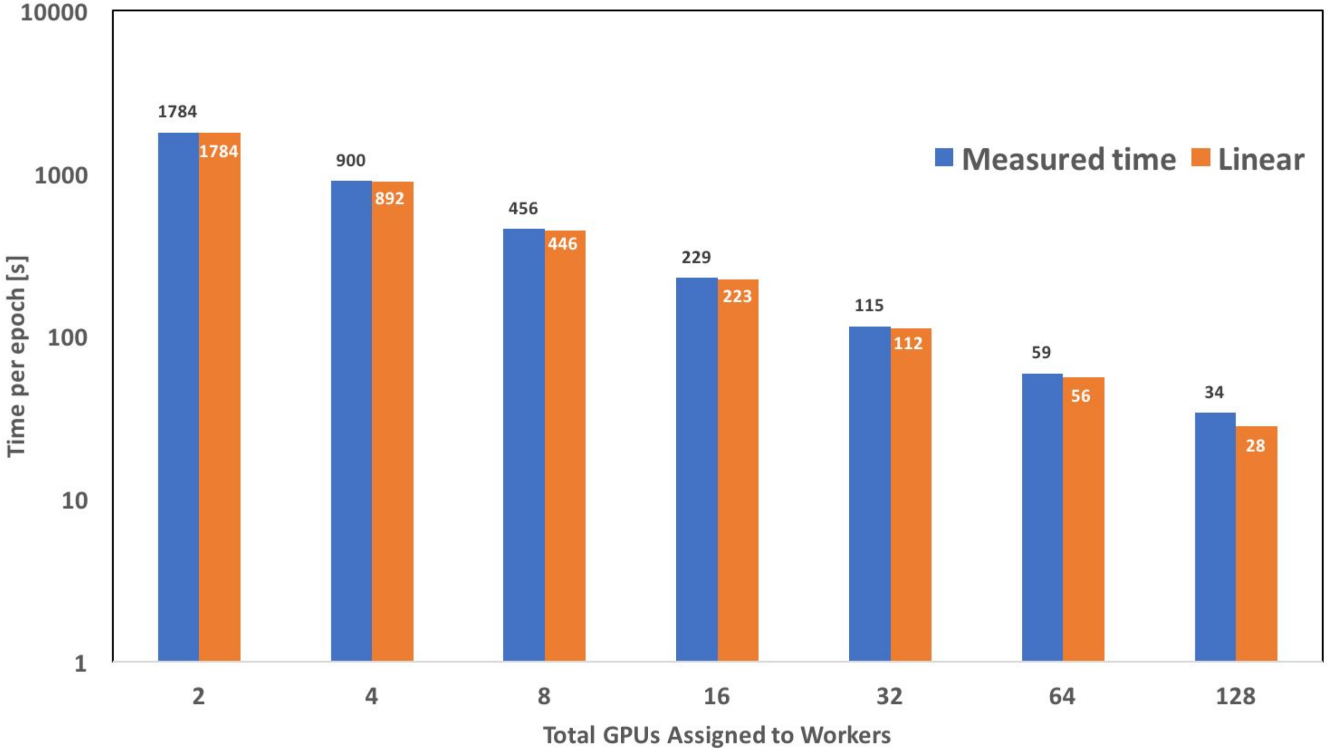
Be able to quickly iterate over different training configurations

Use TFJob

- Test distributed training on a local cluster and on a public cloud
- Rely on 128 (preemptible) Google Cloud GPUs for the distributed training
- Kubeflow cluster running on GKE, deployed with same ArgoCD setup

Results

KubeCon CloudNativeCon Europe 2021 Virtual



Conclusion / Future Steps

Platform available handling all ML lifecycle steps

Improved use of on-premises resources

Ability to scale out to external clouds (GPUs, TPUs, ...)

Ongoing Work

- Onboard new use cases, ex: reinforcement learning for beam calibration
- Provide an easy way for users to curate their environments
  - Binder is a good candidate, looking at integrating with Kubeflow Jupyter
- Improve artifact / metadata versioning and serving

Questions?