Accelerating GAN training using highly parallel hardware on public cloud

Dejan Golubovic

Ricardo Rocha

Renato Cardoso

Sofia Vallecorsa

Ignacio Peluaga Lozada

João Fernandes

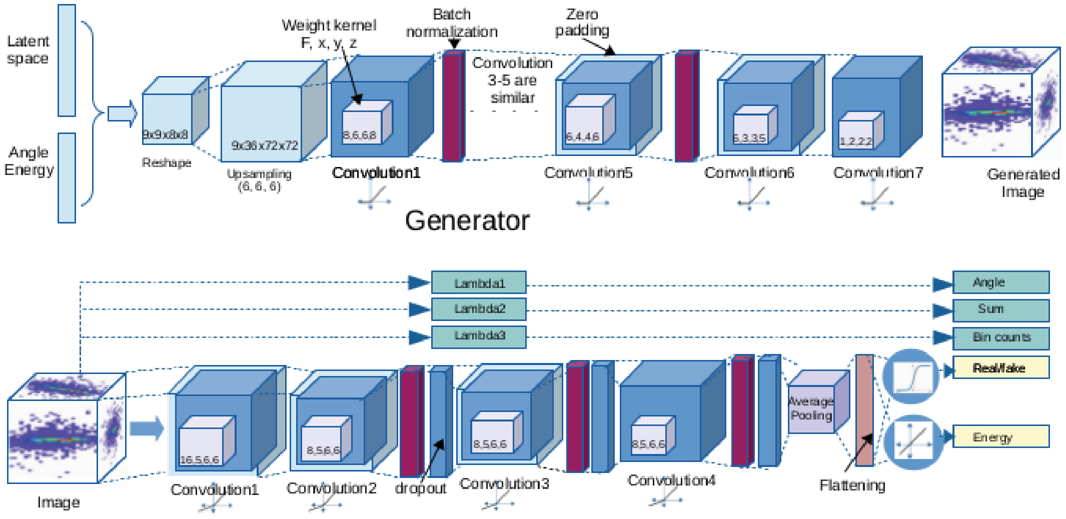
21/04/2021

Introduction

- Objective:
  - Accelerate the training of Deep Learning models, while maintaining the validity of the results
  - Usage of public cloud infrastructure
- Why deploy using public cloud?
  - use of on-demand resources when a longer-term investment on-premises cannot be justified
  - Availability of a large number of GPUs
  - access to specific hardware provided by the cloud vendor (TPU, IPU,)
- TPUs vs GPUs

The Generative Adversarial Network

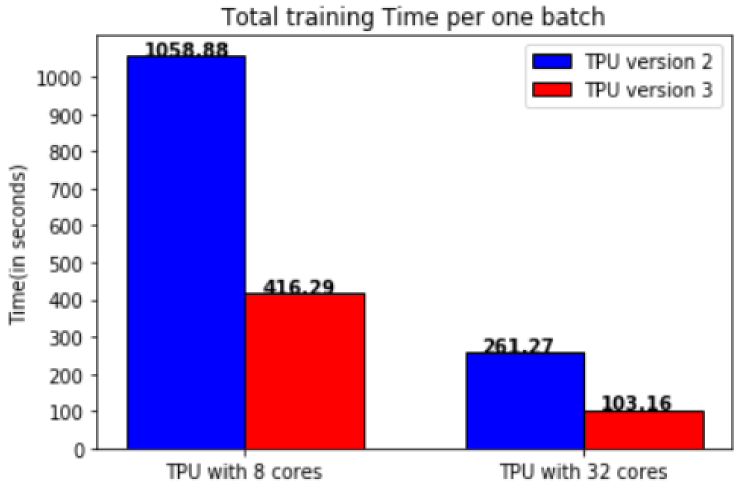
- 3D convolutional Generative Adversarial Network using physics constraints.
- Generates 51x51x25 pixels images, representing energy depositions in calorimeters, similar to the ones generated by Monte Carlo.
- This model was chosen because it is compute bound

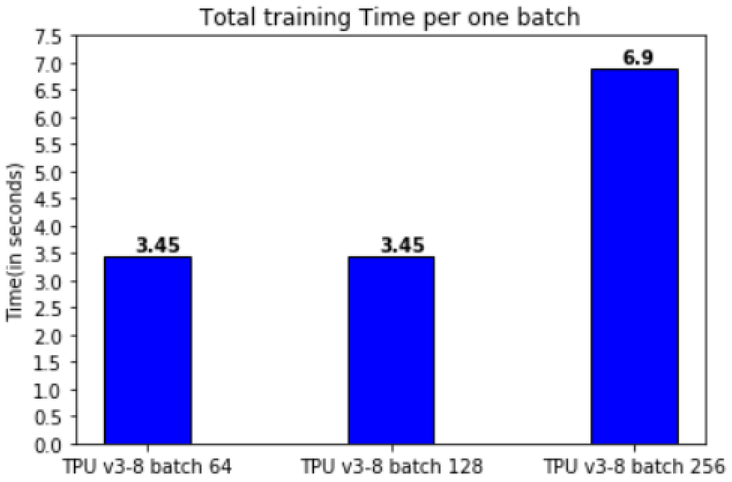


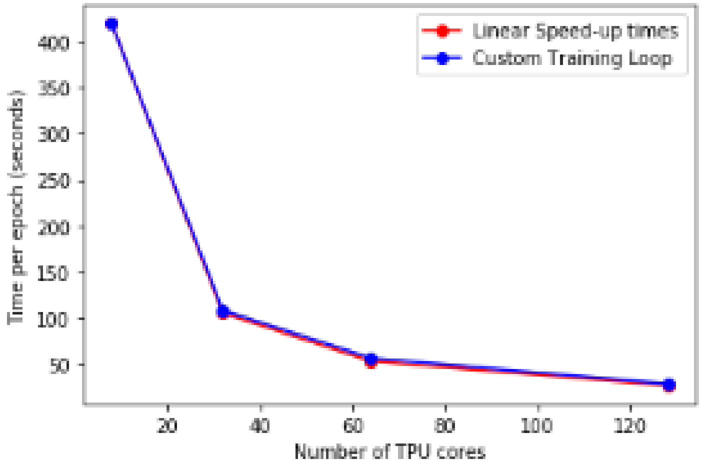
Gulrukh Khattak. "3D convolutional GAN for fast simulation." EPJ Web of Conferences. Vol. 214. EDP Sciences, 2019.

Training on Google TPUs

- Tensor Processing Units (TPUs) are application-specific integrated circuits (ASICs) developed by Google in order to accelerate the machine learning workload.
  - A large two-dimensional matrix multiply unit (MXU) size of 128x128



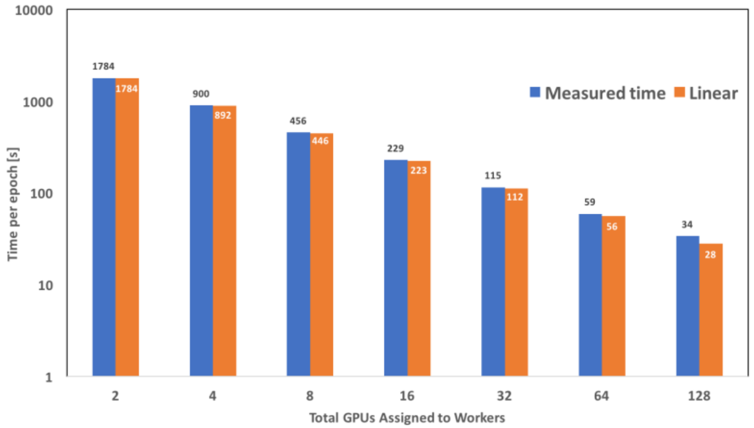


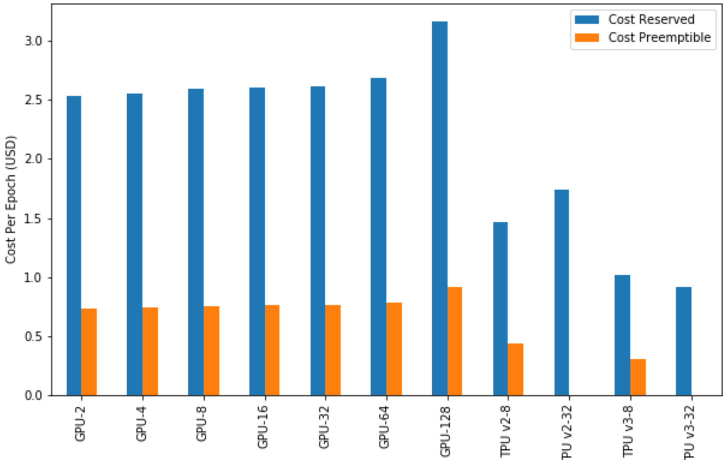


Google Cloud Platform with Kubeflow

Resources access through CloudBank EU

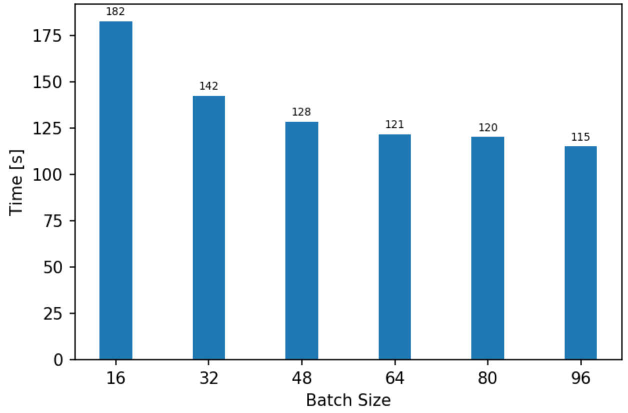
- GPUs configuration
  - The Kubernetes cluster includes multiple node groups, 1 to 8 GPUs per node
  - Tests done using V100 google cloud platform
  - Results show near linear speedup
- Cost Analysis
  - The cost per epoch remains similar when increasing the number of GPUs while the training time is reduced
  - Using preemptible GPUs allows significant cost savings and should be preferred
  - The best results are achieved using preemptible TPU v3-8, which are 2.4 times cheaper than their GPU equivalent.





Google Cloud Platform with Kubeflow

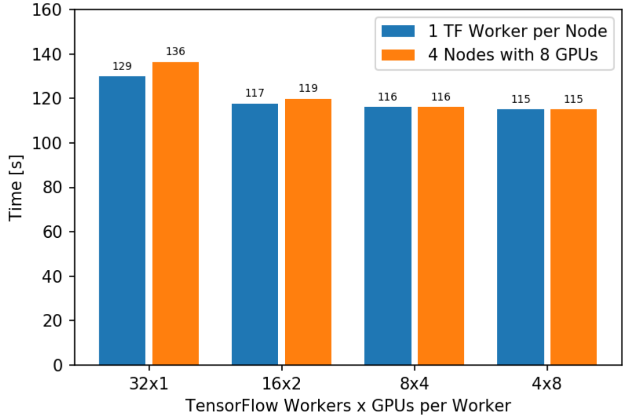
Batch Size Tests



- better generalization performance

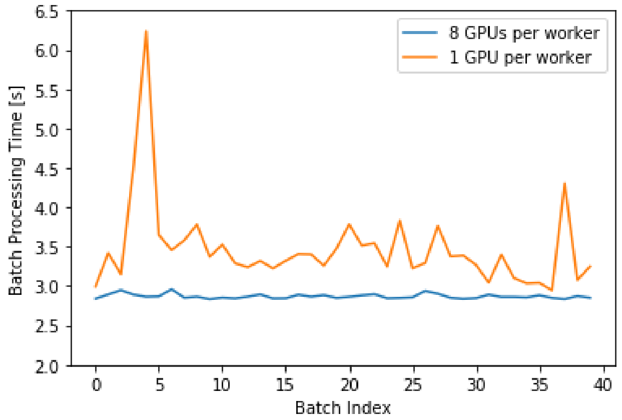
- less steps to complete
- faster training

Cluster Configuration



- optimal configuration
  - more GPUs per node
  - more GPUs per worker
- best results
  - number of workers = number of nodes
  - number of GPUs per worker = the number of GPUs per node

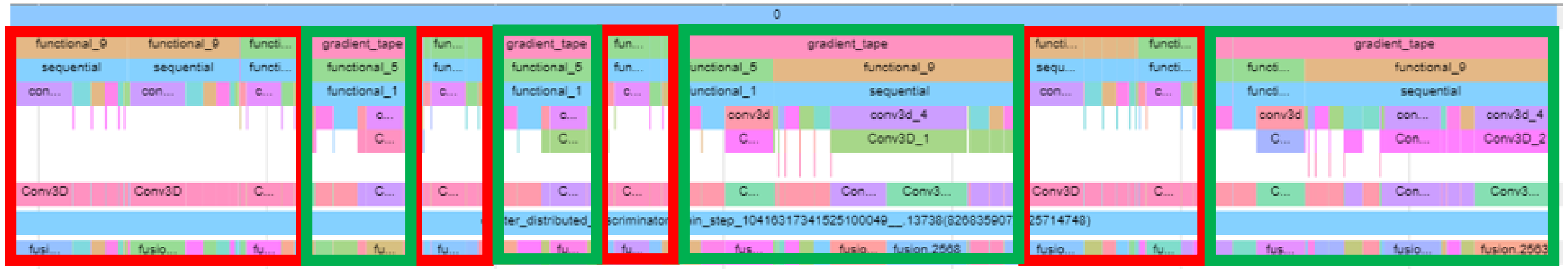
Stability Test



- Sub-optimal configuration makes training time instable and overall longer
- Equal number of GPUs per worker and GPUs per node keeps instability to a minimum

Further Analysis

- TPUs:
  - idle time is 0.7%, with 28.5% for All-Reduce operations
  - 38% for **forward-pass**
  - 61% **for back-propagation**
- GPUs:
  - idle time is 2.9%, mostly for All-Reduce
  - Similar percentages for forward and backward propagation as the TPUs.
- Program is not input bound, 0% of the training step time was spent waiting for input
- With this profile it is possible to verify that the model is compute bound



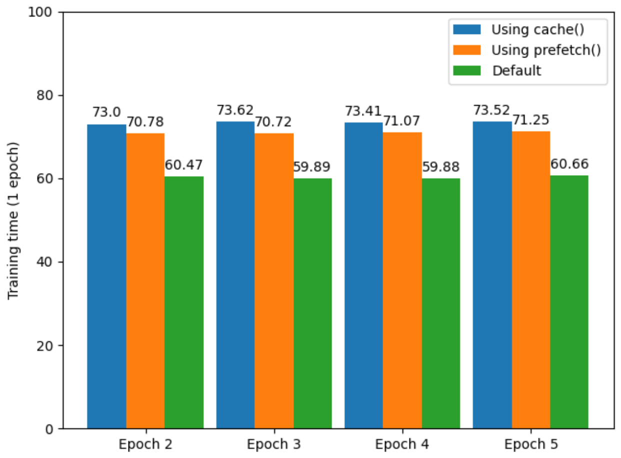
4 gradient tapes

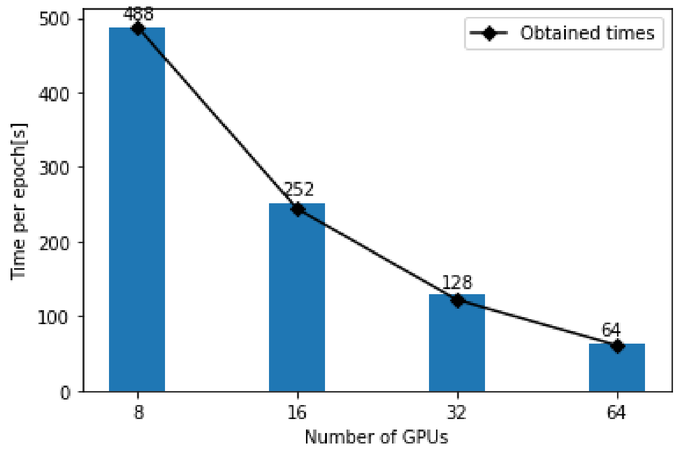
- After each gradient tape there is one all reduce
- Followed by a RMSprop and the corresponding updates

Microsoft Azure with Azure's Machine Learning Service

Resources access through openlab Microsoft Azure Research enrollment

- The 3DGAN deployment uses Azure Machine Learning service
  - Abstracts the user from the underlying infrastructure detail
  - the job needs to be configured while the provisioning of the compute cluster is entirely operationalized by the Azure Machine Learning service.
- GPU-powered nodes with 24 vCPU cores, 448 GiB memory and 4 V100 GPU
- Azure optimizes the data set management according to the hardware infrastructure setup





Conclusions and Future Plans

- Efficient parallelization of the adversarial training process, the 3DGAN training time is brought down from about a week to around one hour.
- Demonstrates the deployment of scientific DL workloads using public cloud services.
- Use case deployment can be optimized in order to reduce costs
- Commoditizing access to public cloud is an efficient way of complementing on premise capacity and also go upper in the stack

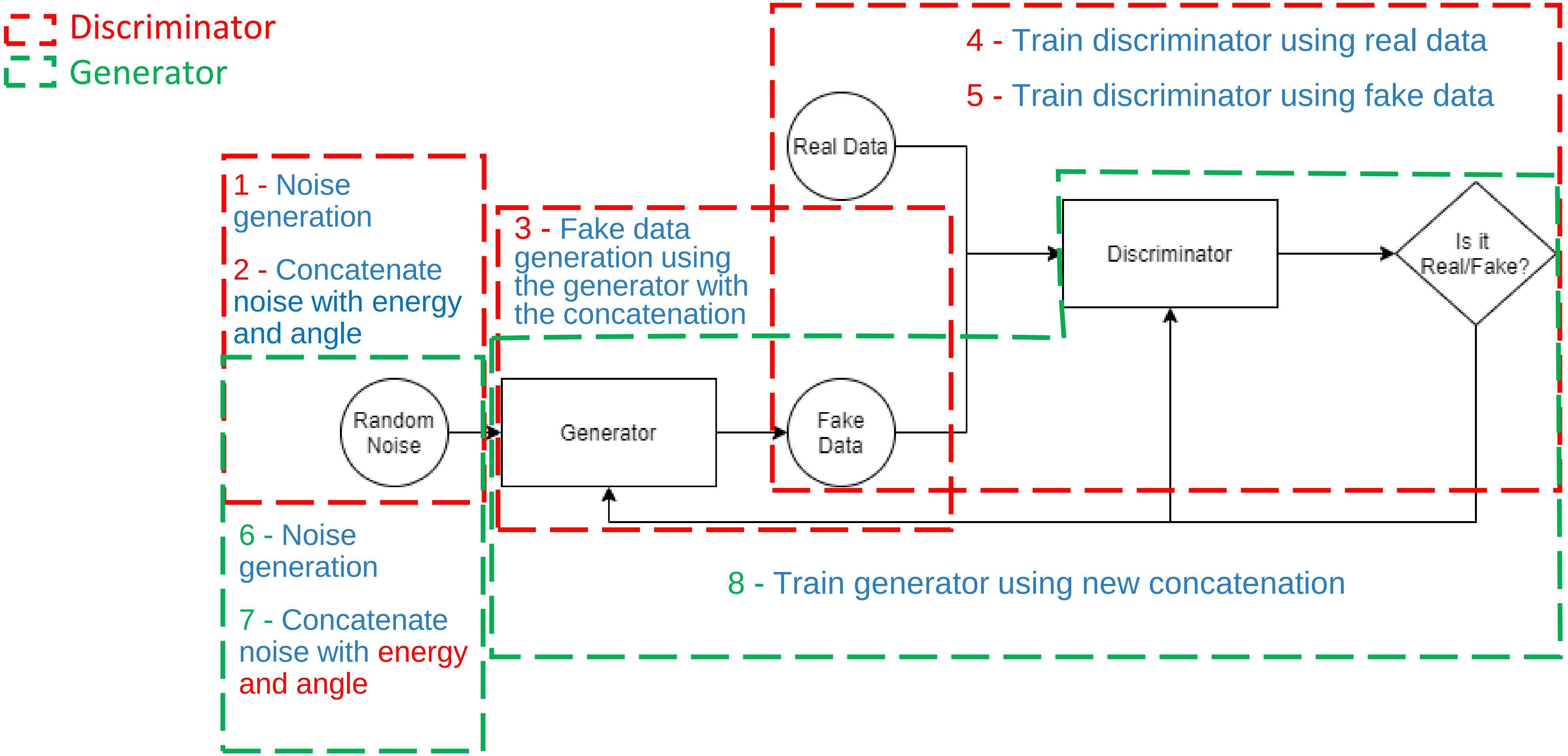
Thank You

QUESTIONS?

renato.cardoso@cern.ch

Sofia.Vallecorsa@cern.ch

Backup - GAN Training

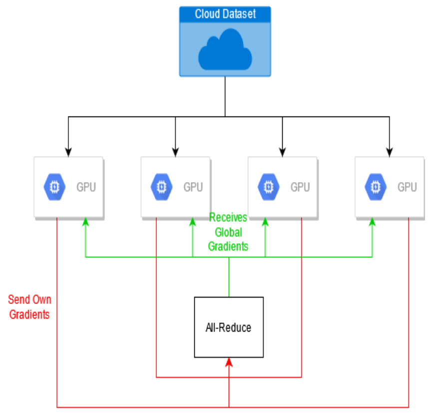


Distributed training in TensorFlow

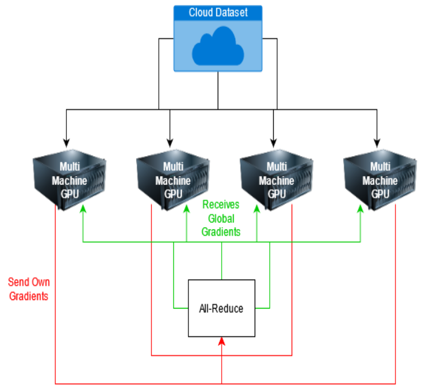
- Achieve distributed training using TensorFlow distributed Strategy tf.distribute.Strategy:
  - A TensorFlow API to distribute training across multiple GPUs, multiple machines with GPUs or TPUs.
- Can be done using a high level TensorFlow training function such as model.fit or by using a custom training loop:
  - Model.fit is not optimal due the GAN architecture

➡ Use a custom training to include all the training steps

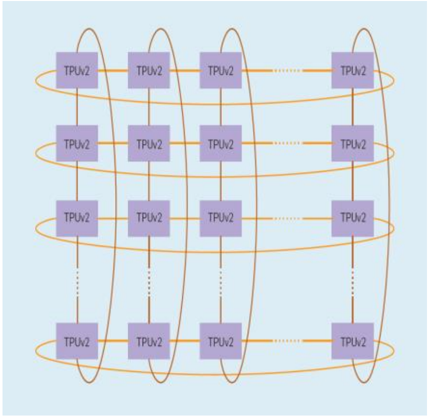
Mirrored Strategy



Multi Worker Mirrored Strategy

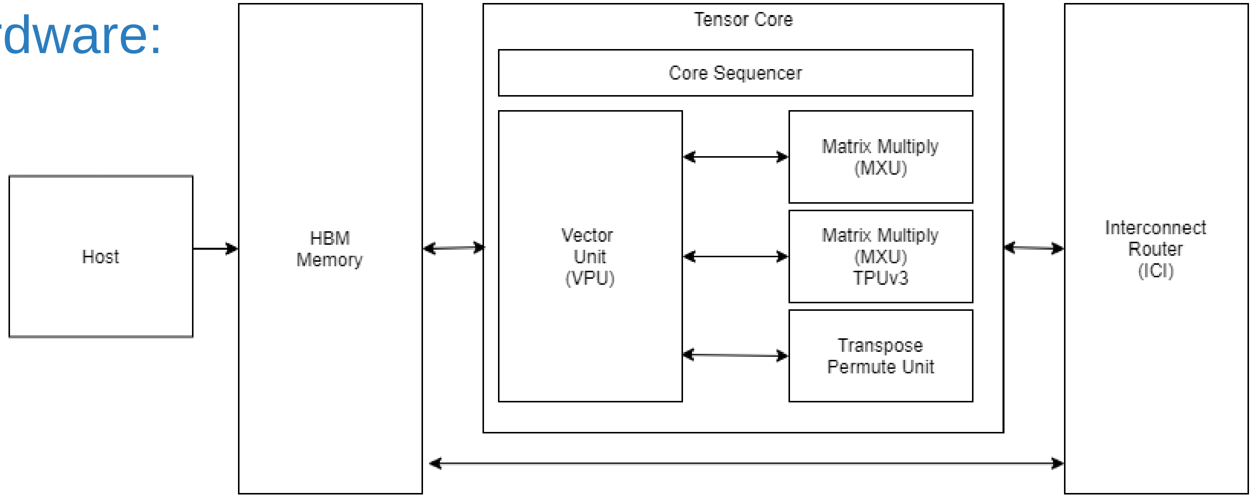


TPU Strategy



Backup - TPUs – Tensor Processing Units

- What are TPUs:
  - Tensor Processing Units (TPUs) are application-specific integrated circuits (ASICs) developed by Google in order to accelerate the machine learning workload.
- What is a TPU hardware:

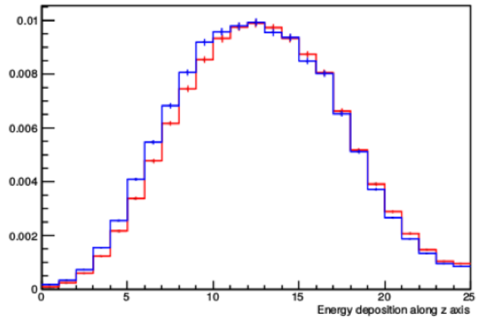


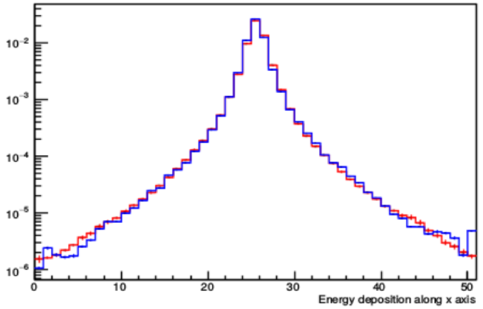
  - A large two-dimensional matrix multiply unit (MXU) size of 128x128

Results

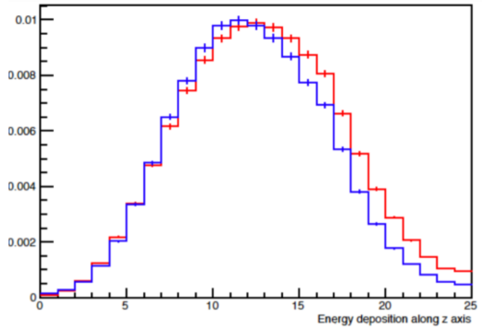
Geant4 GAN

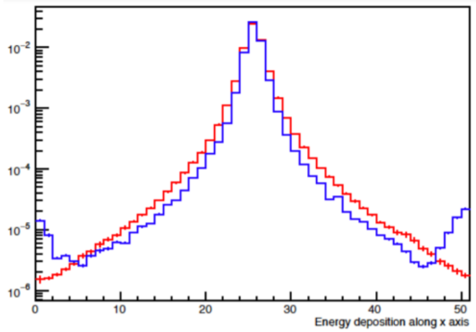
Validation plots obtained from 8 TPU cores





Validation plots obtained from 64 GPU cores





Backup - Code

- Mirrored Strategy

```
mirrored_strategy = tf.distribute.MirroredStrategy()
```

- Multi Worker Mirrored Strategy

```
os.environ["TF_CONFIG"] = json.dumps({"cluster": {
        "worker": ["host1:port", "host2:port", "host3:port]},
        "task": {"type": "worker", "index": 0}})
multiworker_strategy = tf.distribute.experimental.MultiWorkerMirroredStrategy()
```

- TPU Strategy

```
tpu_address = os.environ["TPU_NAME"]
cluster_resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu=tpu_address)
tf.config.experimental_connect_to_cluster(cluster_resolver)
tf.tpu.experimental.initialize_tpu_system(cluster_resolver)
tpu_strategy = tf.distribute.TPUStrategy(cluster_resolver)
```