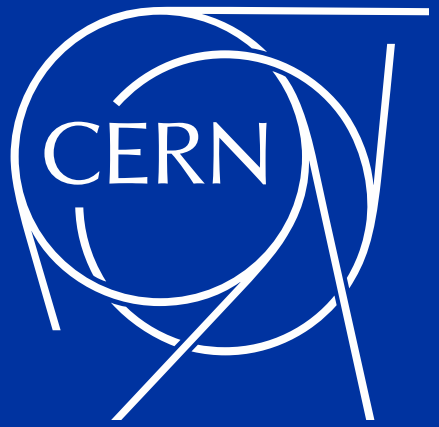


Docker with GPU’s in Batch

Joshua Fenech

Friday 31st January

Outline

- GPU Resources available in Batch
- How these are provisioned
- Monitoring
- Why Docker?
- Nvidia Docker Stack
- Docker Setup
- GPU Docker Setup
- Integrate GPU Docker with HTCondor
- Issues & Future Work

Resources available in Batch

- **16 Nvidia Tesla V100's**
- **Double Precision**
- **32GB DDR3 Memory**
- **"With 640 Tensor Cores, V100 is the world's first GPU to break the 100 teraFLOPS (TFLOPS) barrier of deep learning performance." (Nvidia)**

How these are provisioned

- **4 GPU's installed per server**
  - 32Gb DDR3 Memory
  - 2 x Intel Xeon Silver 4114 CPU's
  - 384GB of DDR4 ECC Reg memory
  - 2x 960GB Micron 5200 Eco SSDs
- **Provisioned via PCI passthrough**
- **Driver installed at the VM level**
- **1 or 4 GPU's per VM currently configured via OpenStack Flavour – no change in driver requirements**
- **1 VM per GPU but >1 GPU per VM**

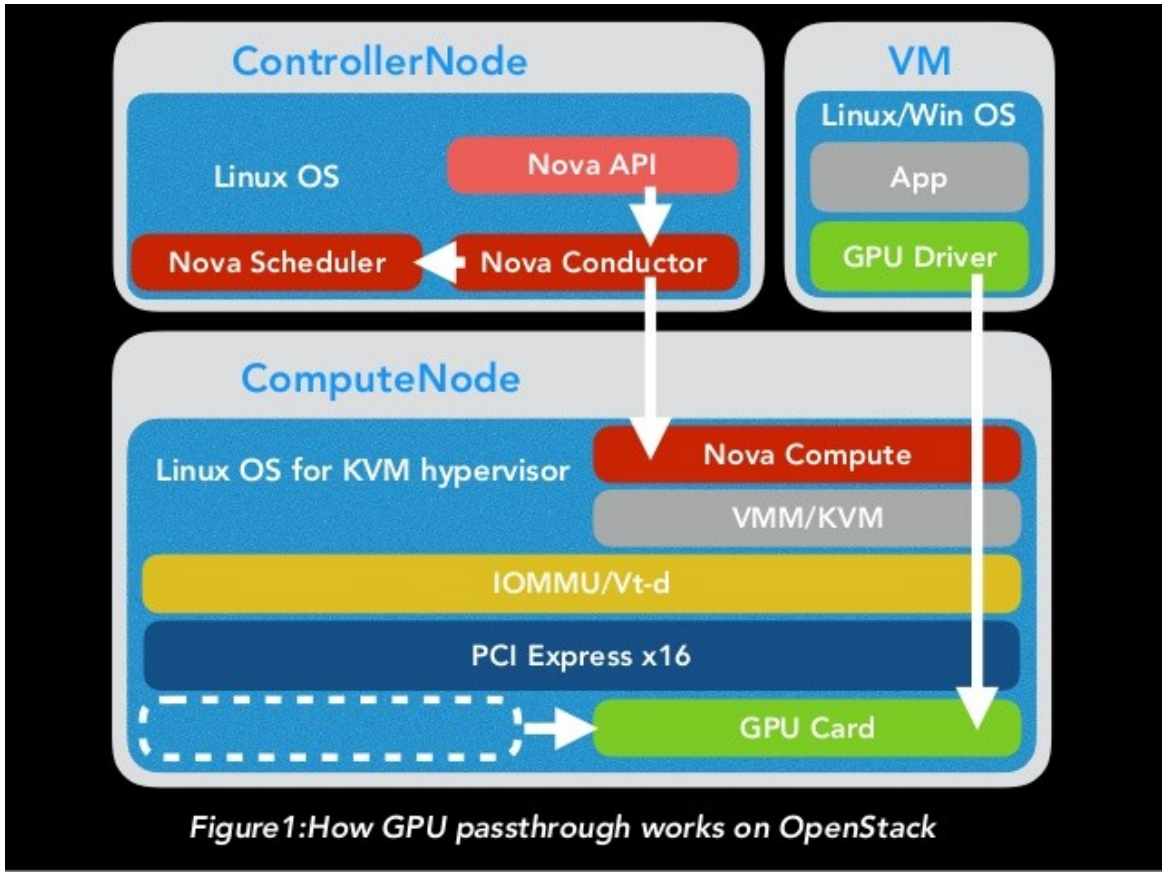


Figure1:How GPU passthrough works on OpenStack

https://www.slideshare.net/LCChina/gpgpu-on-openstack-the-best-practice-for-gpggu-internal-cloud

How these are provisioned

- **Batch GPU's → Scheduling with HTCondor**
  - **Machine ClassAd advertises GPU**
  - **Submit file 'Request_GPUs>0' adds this parameter to JobAd**
  - **Condor matches**

How these are provisioned

```
[jfenech@lxplus723 newdocker]$ cat cpu_docker.py
#!/usr/bin/env python

import sys
import numpy as np
import tensorflow as tf

device_name = "/cpu:0"
with tf.device(device_name):
    random_matrix = tf.random.uniform(shape=[4,4], minval=0, maxval=1)
    dot_operation = tf.matmul(random_matrix, tf.transpose(random_matrix))
    sum_operation = tf.reduce_sum(dot_operation)

with tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(log_device_placement=True)) as session:
        result = session.run(sum_operation)
        print(result)
```

```
[jfenech@lxplus723 newdocker]$ cat gpu.sh
#!/bin/bash
python -m virtualenv myvenv
source myvenv/bin/activate
pip install tensorflow
pip install tensorflow-gpu
python docker.py
```

```
[jfenech@lxplus723 newdocker]$ cat gpu.sub
universe                = vanilla
executable              = ./gpu.sh
arguments               = /etc/hosts
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
output                  = out/$(ClusterId).$(ProcId).out
error                   = err/$(ClusterId).$(ProcId).err
log                     = log/$(ClusterId).$(ProcId).log
+JobFlavour             = "espresso"
request_gpus            = 1
queue 1
```

How these are provisioned

```
[jfenech@lxplus723 newdocker]$ cat gpu_docker.py
#!/usr/bin/env python

import sys
import numpy as np
import tensorflow as tf

device_name = "/gpu:0"
with tf.device(device_name):
    random_matrix = tf.random.uniform(shape=[4,4], minval=0, maxval=1)
    dot_operation = tf.matmul(random_matrix, tf.transpose(random_matrix))
    sum_operation = tf.reduce_sum(dot_operation)

with tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(log_device_placement=True)) as session:
        result = session.run(sum_operation)
        print(result)
```

```
[jfenech@lxplus723 newdocker]$ cat gpu_docker.sub
universe                = docker
docker_image            = tensorflow/tensorflow:1.15.0-gpu
executable              = ./gpu_docker.py
arguments               = /etc/hosts
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
output                  = out/$(ClusterId).$(ProcId).out
error                   = err/$(ClusterId).$(ProcId).err
log                     = log/$(ClusterId).$(ProcId).log
+JobFlavour             = "espresso"
request_gpus            = 1
requirements            = Machine == "b7g47n0009.cern.ch"
queue 1
```

Monitoring

- GPU’s monitored at VM level via Collectd-cuda plugin now in Production
- Collects information via Nvidia Service Management Interface
- Current month’s logs visible in Cuda Monitoring Grafana dashboard
- Logs stored in S3 available via Spark in SWAN
- Metrics available include temperature, utilisation, clocks, power
- Later HTCondor versions (>8.8) include enhanced per job GPU monitoring
  - Average Usage
  - Peak Memory Usage

Monitoring

```
[root@b7g47n0008 ~]# nvidia-smi
Thu Jan 30 16:58:35 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  Off  | 00000000:00:05.0 Off |                    0 |
| N/A   40C    P0    27W / 250W |     12MiB / 32510MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Utilisation

Percent of time over the past sample between 1 second and 1/6 second period during which one or more kernels was executing on the GPU

Monitoring

```
[root@b7g47n0008 ~]# nvidia-smi
Thu Jan 30 16:58:35 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name
| Fan  Temp  Pe
|===============
|   0  Tesla V1
| N/A   40C
+--------------

+--------------
| Processes:
|  GPU       PI
|===============
|  No running p
+--------------
```

Utilisation

Percent of time over the past sample between 1 second and 1/6 second period during which one or more kernels was executing on the GPU

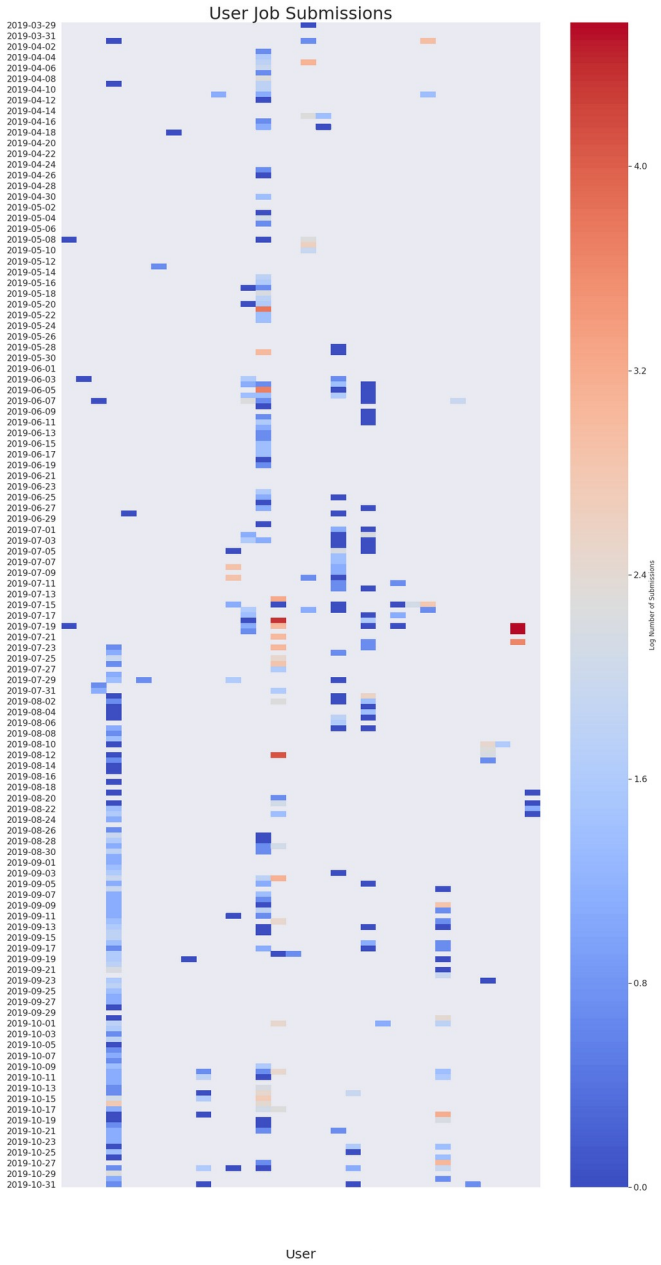
https://monit-grafana.cern.ch/d/0EfSgd4Zz/cuda-monitoring?orgId=5&from=now%2FM&to=now

Monitoring

```
[root@b7g47n0008 ~]# nvidia-smi
Thu Jan 30 16:58:35 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name
| Fan  Temp  Pe
|===============
|   0  Tesla V1
| N/A   40C
+--------------

+--------------
| Processes:
|  GPU       PI
|===============
|  No running p
+--------------
```

https://monit-grafana.cern.ch/d/0EfSgd4Zz/cuda-monitor
orgId=5&from=now%2FM&to=now



Why Docker?

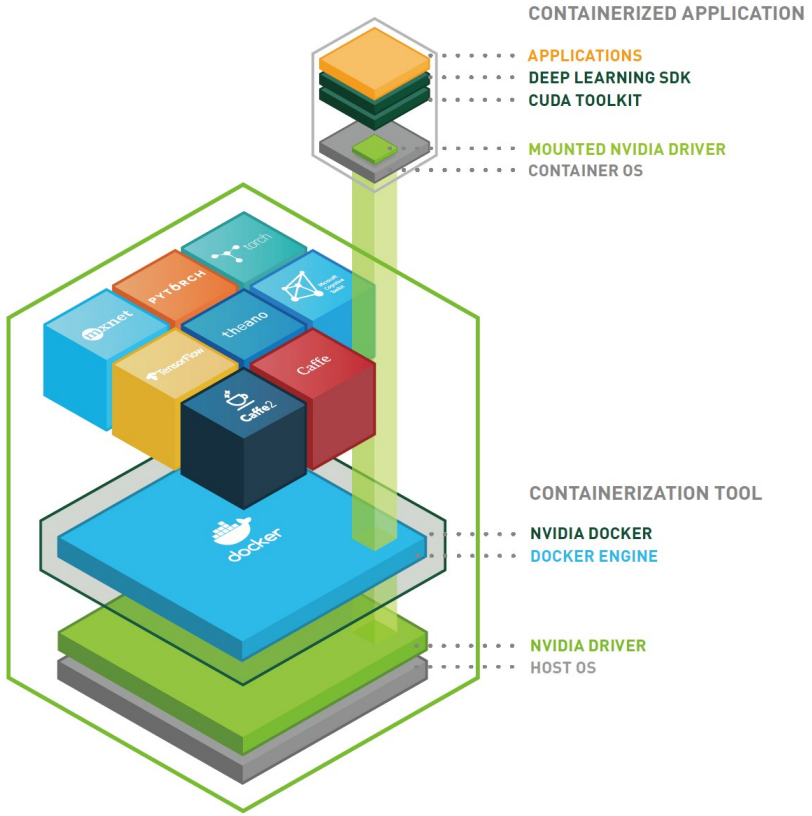
- **Reproducible environment independent of host**
- **Let user manage requirements and dependencies**
- **Variety of GPU use-cases and frameworks**

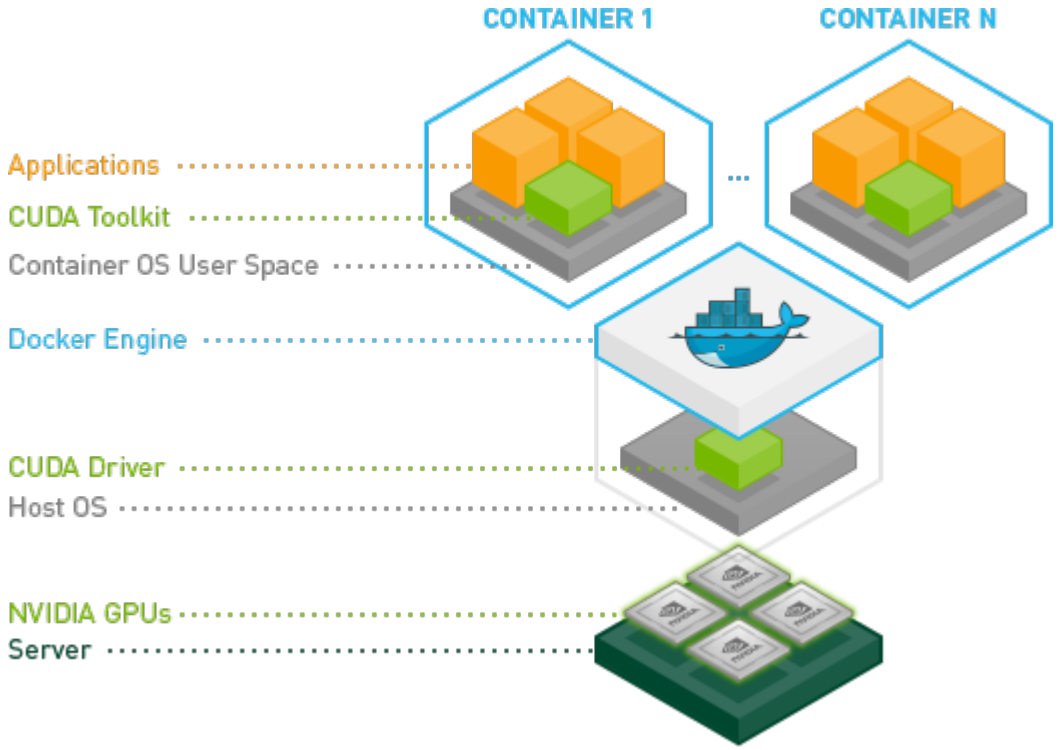
Why Docker?

- **Benchmarking!**

NVIDIA Docker Structure

Nvidia ≡ Cuda





https://dgx-wiki.readthedocs.io/en/latest/docs/docker/service.html#nvidia-docker

https://www.slideshare.net/LCChina/gpgpu-on-openstack-the-best-practice-for-gpggu-internal-cloud

Setup GPU Docker

- Install libnvidia-container, nvidia-container-runtime, nvidia-container-runtime-hook
- GPU integration is better in later versions of Docker
- CentOS repo ships with Docker version: 1.13
- Latest Docker Version: 19.03
- → Upgrade Docker
- Docker → Docker-ce
- Docker-ce requires containerd.io, which conflicts with earlier Docker versions
- → Remove Docker → Install containerd.io, docker-ce, docker-ce-cli
- Add Docker to Condor user group to give required permissions

Integrate GPU Docker with HTCondor

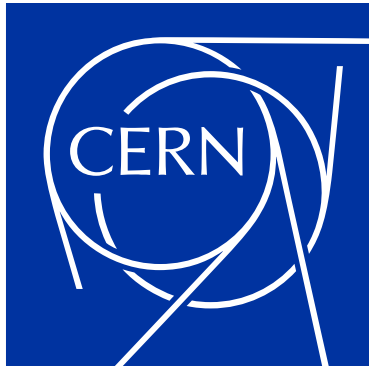
- **Docker requires 2 flags/parameters in order to run on GPU, “--gpus all”**
  - **docker run –gpus all tensorflow/tensorflow:latest-gpu**
- **HTCondor 8.9.1 introduced greater GPU Docker integration:**
- **GPU’s mounted automatically in (GPU + Docker) jobs**
- **DOCKER_EXTRA_ARGUMENTS flag allows the crucial ‘--gpus all’ to be added to the docker create command that Condor utilises**
- **Improved GPU job monitoring with the additional metrics (average usage, peak memory usage)**
- **Upgrade HTCondor (from default 8.8.6) on GPU machines**
- **Requires dev repo**
- **Also need to ensure Machine ClassAd contains “HasDocker” flag**

Issues & Future Work

- **Varied workloads but single hardware**
  - **Persistent low utilisation**
  - **Some algorithms (e.g. LSTM's) struggle to make full use of GPU**
  - **Making full use of Tensor Cores depends on user providing precisely shaped inputs (well, divisible by 8 (for FP16 data) or 16 (INT8 data)) to match the GPU architecture**
- **Fairsharing limited number of resources**
  - **Many jobs are long (nextweek), blocking larger number of short jobs**
- **Can Docker manage GPU sharing??**
  - **Harness slack GPU resources**
- **Machine learning vs SixTrack workloads**

Batchdocs GPU Tutorial

Grafana GPU Dashboard



home.cern