Singularity GPU Containers Options

A. Gutcaits, Dr. chem.
2021-03-05

GPU Singularity Containers

NGC Catalog

- NGC (NVIDIA GPU-Accelerated Containers) offers a comprehensive catalog of GPU-accelerated software for Deep Learning, HPC and Visualization applications.
- Consists of containers, pre-trained models, Helm charts for Kubernetes deployments and industry specific AI toolkits with SDK
- Guest Access and Authenticated Access
- **GPUs compatible with NGC: V100, A100, T4, Jetson, RTX Quadro**.

Docker Hub

- Official Docker images for the machine learning framework
- Containers with GPU support marked "-gpu"
- Usually, Guest Access

For Future reading: NVIDIA Cloud Documentation, List of Nvidia graphics processing units

GPU Compute Capability (NGC)

Compute Capability

- The **compute capability** of a device is represented by a version number, also sometimes called its "**SM version**". This version number identifies the **features supported by the GPU hardware.** The compute capability comprises a major revision number X and a minor revision number Y and is denoted by **X.Y**

- Note: **The compute capability version** of a particular GPU **should not be confused with the CUDA version** (e.g., CUDA 7.5, CUDA 8, CUDA 10.2 etc), which is the version of the **CUDA software platform**. The CUDA platform is used by application developers to create applications that run on many generations of GPU architectures. New versions of the CUDA platform typically also include software features that are independent of hardware generation.

	Fermi†	Kepler†	Maxwell‡	Pascal	Volta		
	sm_20	sm_30	sm_50	sm_60	sm_70	======>	**Tesla V100**
Tesla K20, K40 ======>		sm_35	sm_52	sm_61	sm_72		
		sm_37	sm_53	sm_62			

For Future reading: CUDA Programming_Guide, CUDA-Enabled GPUs, NVIDIA Deep Learning Frameworks

Running NGC Containers with Singularity (1)

- **Register and Activate Your NGC Account**
- **Generate Your NGC API Key**
- **Load singularity module:**

$ module load singularity/3.2.1
$ module list
Currently Loaded Modules:
singularity/3.2.1

- **Setting NGC container registry authentication credentials:**

$ export SINGULARITY_DOCKER_USERNAME='$oauthtoken'
$ export SINGULARITY_DOCKER_PASSWORD=<NVIDIA NGC API key>

- **Pulling container to a local Singularity image:**

$ export SINGULARITY_TMPDIR=/home/TMP/$USER
$ mkdir -p $SINGULARITY_TMPDIR
$ singularity pull tensorflow_19.06-py3.sif docker://nvcr.io/nvidia/tensorflow:19.06-py3

- **For containers from Docker Hub**

$ singularity pull docker://tensorflow/tensorflow:2.3.1-gpu

- **GPU use:**

*If your host system has an NVIDIA GPU card and a driver installed, you can leverage the card with the **--nv** option*

For Future reading: Singularity documentation, User Guide, Quick Start, Deep Learning Support Matrix

Running NGC and GPU Containers with Singularity (2)

Checking TensorFlow version

```
$ singularity exec --nv  tensorflow_19.06-py3.sif python -c 'import tensorflow as tf; print(tf.__version__)'
1.13.1
$ singularity exec --nv tensorflow_2.3.1-gpu.sif python -c 'import tensorflow as tf; print(tf.__version__)'
2.3.1
```

Checking available GPUs (Interactive shell)

```
@ui-1$ qsub -I -l nodes=1:ppn=2:gpus=1                    ===> For getting node with GPU
@ui-1$  qsub -I -l nodes=1:ppn=2:gpus=1 -l feature=v100   ===> For getting node with Tesla V100 GPU
qsub: waiting for job 1028957.rudens to start
qsub: job 1028957.rudens ready
@wn60 $ cd $PBS_O_WORKDIR
@wn60 $ module load singularity
$ singularity exec --nv  tensorflow_19.06-py3.sif tf_gpu_avail_01.py
…
['/device:CPU:0', '/device:XLA_CPU:0', '/device:XLA_GPU:0', '/device:XLA_GPU:1', '/device:GPU:0', '/device:GPU:1']
```

Python script tf_gpu_avail_01.py

```
import tensorflow as tf
import os
from tensorflow.python.client import device_lib
print([device.name for device in device_lib.list_local_devices() if device.name != None])
```

Running NGC and GPU Containers with Singularity(3)

Running simple TensorFlow script (tf2_gpu_cpu.py)

```
import tensorflow as tf
import os
import time
cpu_slot = 0
gpu_slot = 0

# Using CPU at slot 0
with tf.device('/CPU:' + str(cpu_slot)):
    # Starting a timer
    start = time.time()
    A = tf.constant([[3, 2], [5, 2]])
    print(tf.eye(2,2))
# Printing how long it took with CPU
    end = time.time() - start
    print(end)

# Using the GPU at slot 0
with tf.device('/GPU:' + str(gpu_slot)):
    # Starting a timer
    start = time.time()
    # Doing operations on CPU
    A = tf.constant([[3, 2], [5, 2]])
    print(tf.eye(2,2))

    # Printing how long it took with GPU
    end = time.time() - start
    print(end)
    print(('Your TensorFlow version: {0}').format(tf.__version__))
```

This script is comparing time spent using the CPU versus GPU. The simple operation here is creating a constant and an identity matrix by defining a tensor A, getting the rows and columns and making an identity matrix.

```
@ui-2 tf_gpu]$ qsub -I -l nodes=1:ppn=2:gpus=1
qsub: waiting for job 1045525.rudens to start
qsub: job 1045525.rudens ready
@wn56 ~]$  cd $PBS_O_WORKDIR
@wn56 ~]$ module load singularity
@wn56 tf_gpu]$ singularity exec --nv tensorflow_2.3.1-gpu.sif python
tf2_gpu_cpu.py
...
tf.Tensor(
[[1. 0.]
 [0. 1.]], shape=(2, 2), dtype=float32)
0.006877899169921875              ===>  GPU time
tf.Tensor(
[[1. 0.]
 [0. 1.]], shape=(2, 2), dtype=float32)
0.13370490074157715               ===>  CPU time
```

Notes on Running NGC Containers with Singularity

- **If your host system has an NVIDIA GPU card and a driver installed, you can leverage the card with the --nv option**
- **There is currently a bug in Singularity 3.1.x and 3.2.x causing the LD_LIBRARY_PATH to be incorrectly set within *some container's* environment. As a workaround the LD_LIBRARY_PATH must be unset before invoking Singularity:**

```
$ LD_LIBRARY_PATH="" singularity exec
```

- **There is not necessary to load cuda module due to containers already has installed CUDA environment. In some cases, may be necessary to set for compatibility issues $CUDA_HOME variable by loading module cuda, or searching for other versions:**

```
$ module spider cuda
Versions:
    cuda/cuda-7.5
    cuda/cuda-8.0
    cuda/cuda-9.2
    cuda/cuda-10.1
    cuda/cuda-10.2
```

Main Singularity Commands

Main online container registries and commands to load

Singularity Library:	https://cloud.sylabs.io/library :	$ singularity pull library://
Docker Hub:	https://hub.docker.com :	$ singularity pull docker://
Singularity Hub:	https://singularity-hub.org :	$ singularity pull shub://
NVIDIA GPU Cloud:	https://ngc.nvidia.com:	$ singularity pull docker://nvcr.io/

```
$ singularity pull docker://gcc:5.3.0
$ singularity pull library://godlovedc/demo/lolcow
$ singularity pull docker://nvcr.io/nvidia/tensorflow:19.06-py3
```

Image examinations

```
$ singularity verify lolcow_latest.sif          # Verify signatures
$ singularity inspect  lolcow_latest.sif        # To show labels
$ singularity inspect -d lolcow_latest.sif      # To show the Singularity recipe file (deffile)
$ singularity inspect -r lolcow_latest.sif      # To inspect the runscript
```

Singularity cache

```
$ HOME/.singularity/cache/                      # User's Singularity cache directory
$ singularity cache list                        # List and size of cached files
$ singularity cache clean -a                    # clean all the cache
$ singularity cache clean --type=library,oc     # clean only library, and oci cache
```

Run Command

*Run **defined set of commands from a definition file's runscript**. It is only available when using an image that was built from a definition file that specified a runscript.*

```
$ singularity run library://godlovedc/demo/lolcow
$ singularity run  ./lolcow_latest.sif
```

Exec command

Executes command from container

```
$ singularity exec  ./lolcow_latest.sif  fortune
$ singularity exec  library://godlovedc/demo/lolcow  fortune
```

Shell command

The shell command allows you to spawn a new shell within your container and interact with it as though it were a small virtual machine

```
$ singularity shell library://godlovedc/demo/lolcow
$ singularity shell container.sif
```

User-defined bind paths: $ singularity shell --bind /scratch, src[:dest[:opts]], container.sif

GPU use: *If your host system has an NVIDIA GPU card and a driver installed, you can leverage the card with the **--nv** option :*

```
$ singularity run --nv  container.sif
```

For Future reading: Singularity documentation, User Guide, Quick Start

ARCHIVE