Using Tensorboard Profiler for ML Workload on GPU

This doc will guide you through the process of using Tensorboard Profiler to profile and inspect machine learning workloads on GPU.

Step 1: Set Up Your Environment

Before starting with Tensorboard Profiler, ensure that your environment is ready by following these steps:

Check GPU Availability: Use `nvidia-smi` to confirm that your system has a GPU available and properly set up.

```
sh> nvidia-smi

+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.183.01             Driver Version: 535.183.01   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Quadro P1000                   On  | 00000000:65:00.0 Off |                  N/A |
| 34%   29C    P8              N/A /  N/A |      2MiB /  4096MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

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

Create a Python Virtual Environment: It's a good practice to use a virtual environment to avoid conflicts between dependencies.

```
python3 -m venv jaxvenv
source jaxvenv/bin/activate
```

Install Required Packages: Install TensorFlow (for TensorBoard) and JAX, ensuring that JAX is set up with GPU support.

```
pip install tf-nightly tbp-nightly
pip install --upgrade "jax[cuda12_pip]" -f https://storage.googleapis.com/jax-releases/jax_cuda_releases.html
```

Verify JAX GPU Backend: After installation, verify that JAX is configured to use the GPU backend.

```
python -c "import jax; print(f'Jax backend: {jax.default_backend()}')"
```

Step 2: Create and Run a Test Script

Next, you'll create a simple script to test the profiling setup. This script sets up a basic JAX computation and profiles it using Tensorboard Profiler. Once the script is created, you can execute it to generate profiling data.

```
cat << EOF > test.py
import jax
from jax import numpy as jnp

@jax.named_call
def foo(x, y):
  return (x + y) / 2.

@jax.jit
def bar(a):
  def foo2(x, y):
    return foo(x, y), None

  out, _ = jax.lax.scan(foo2, 0., a)
  return out

a = jnp.array([1., 2., 3., 4., 5.])

print(jax.devices())
jax.profiler.start_trace('/tmp/tensorboard')
with jax.profiler.StepTraceAnnotation('step', step_num=0): # JIT warm-up
  out = bar(a)
with jax.profiler.StepTraceAnnotation('step', step_num=1):
  out = bar(a)
out.block_until_ready()
jax.profiler.stop_trace()
EOF

python3 test.py
```

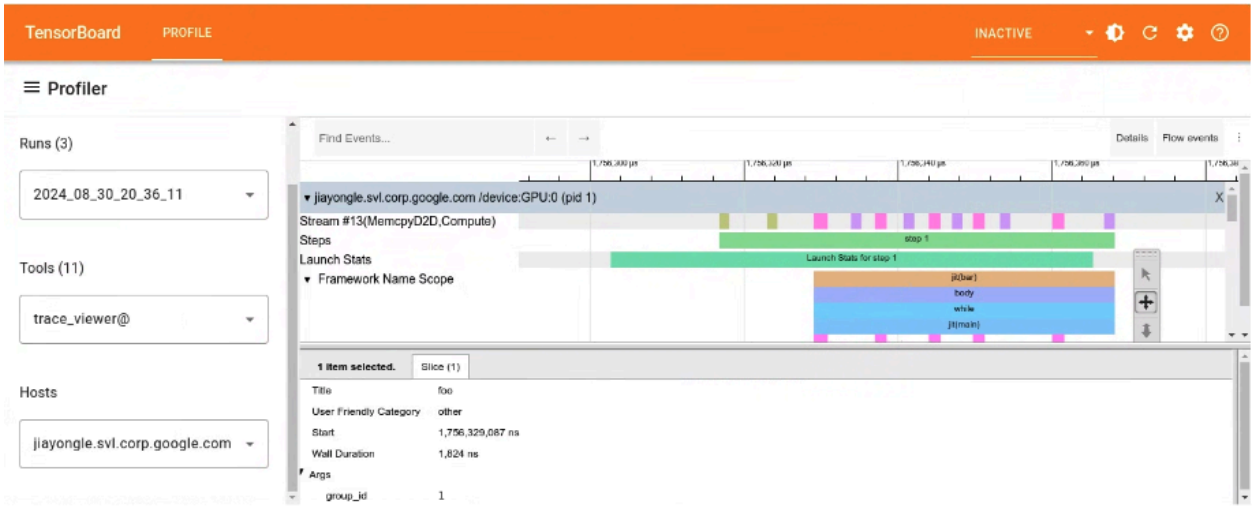
Step 3: Visualize the Profiling Data with TensorBoard

Now that you have the profiling data, you can visualize it using TensorBoard.

1. Launch TensorBoard: Start TensorBoard and point it to the directory where the profiling data was saved.

```
sh> tensorboard --port 6006 --logdir /tmp/tensorboard
```

2. Access TensorBoard: Open your web browser and navigate to http://localhost:6006 to see the profiling results.



Step 4: Run an Additional Example

For a more complex example, you can run a script that uses the MNIST dataset.

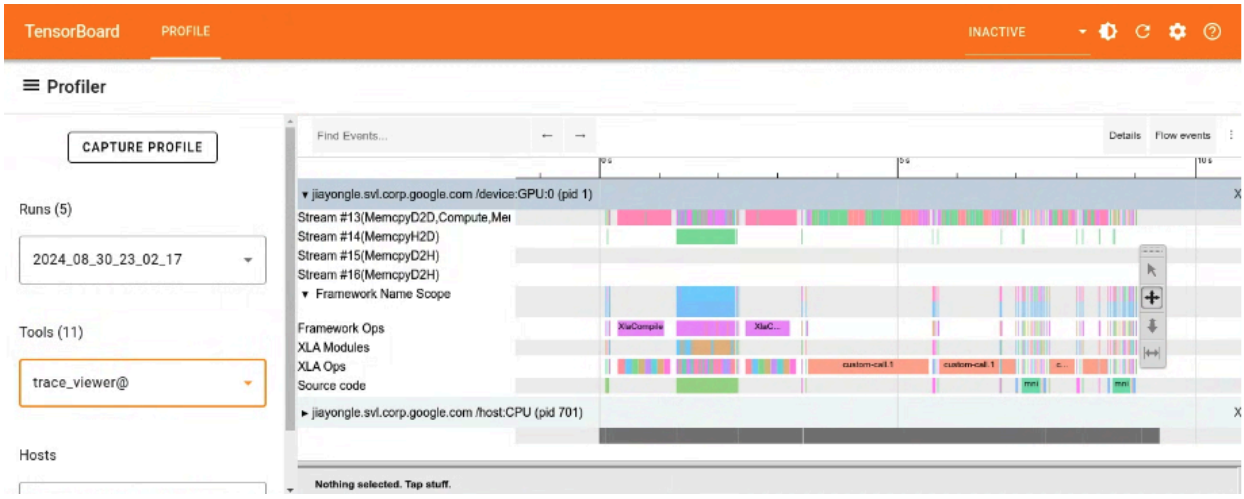
1. Install Additional Dependencies: Install TensorFlow datasets, which include the MNIST dataset.

```
pip install tensorflow_datasets
```

2. Download and Run the MNIST Script: Download the mnist.py script from the provided GitHub link and run it.

```
wget https://raw.githubusercontent.com/jiayongle/colabs/main/python/mnist.py
python3 mnist.py
```

3. Access TensorBoard: Open your web browser and navigate to http://localhost:6006 to see the profiling results.


TensorBoard
PROFILE
INACTIVE
Profiler
CAPTURE PROFILE
Runs (5)
2024_08_30_23_02_17
Tools (11)
trace_viewer@
Hosts
Find Events...
Details
Flow events
▾ jiayongle.svl.corp.google.com /device:GPU:0 (pid 1)
Stream #13(MemcpyD2D,Compute,Mer
Stream #14(MemcpyH2D)
Stream #15(MemcpyD2H)
Stream #16(MemcpyD2H)
▾ Framework Name Scope
Framework Ops
XLA Modules
XLA Ops
Source code
▸ jiayongle.svl.corp.google.com /host:CPU (pid 701)
Nothing selected. Tap stuff.