Profiling Machine Learning Code on HPC Clusters

Oliver Cairncross

Research Computing Centre

Who Needs This

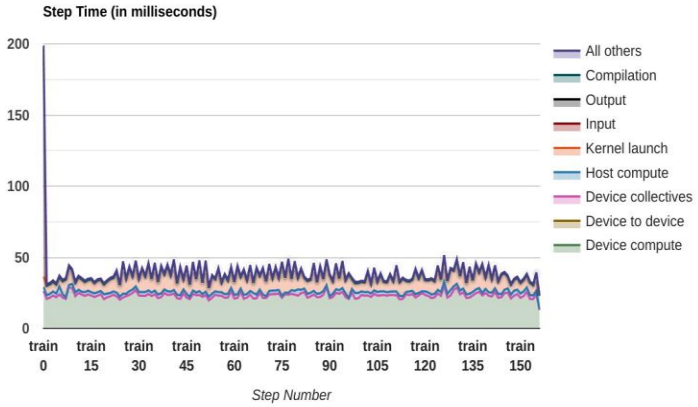
- General Researchers
  - Have a need for machine learning but aren't experts.
  - Wanting to deploy existing solutions but have problems with (non) performance when deploying on our platform.
  - Find that a solution can't handle bigger datasets or too slow.
- Expert Users
  - Need to deal with complex issues.
  - Might need some pointers on how to tackle problems on a HPC environment
- System Administrators
  - Want to test that infrastructure is deployed / configured effectively

Profiling Methods

- Use the default built in platform profiler (Tensorflow, PyTorch, etc).
  - This typically involves turning it on in code and viewing the generated data with a GUI.
  - Can point out issues that stand out
  - Provide nice overview of what your program is doing (e.g., is the GPU being kept busy)
  - Often when profiling code that processes large datasets; the profiler will generate its own huge impractical dataset.

Summary from TensorFlow profiler

Step-time Graph



Recommendation for Next Step

- Your program is NOT input-bound because only 2.8% of the total step time sampled is waiting for input. Therefore, you should focus on reducing other time.
- 29.9 % of the total step time sampled is spent on 'Kernel Launch'. It could be due to CPU contention with tf.data. In this case, you may try to set the environment variable TF_GPU_THREAD_MODE=gpu_private.
- Only 0.0% of device computation is 16 bit. So you might want to replace more 32-bit Ops by 16-bit Ops to improve performance (if the reduced accuracy is acceptable).

Tool troubleshooting / FAQ

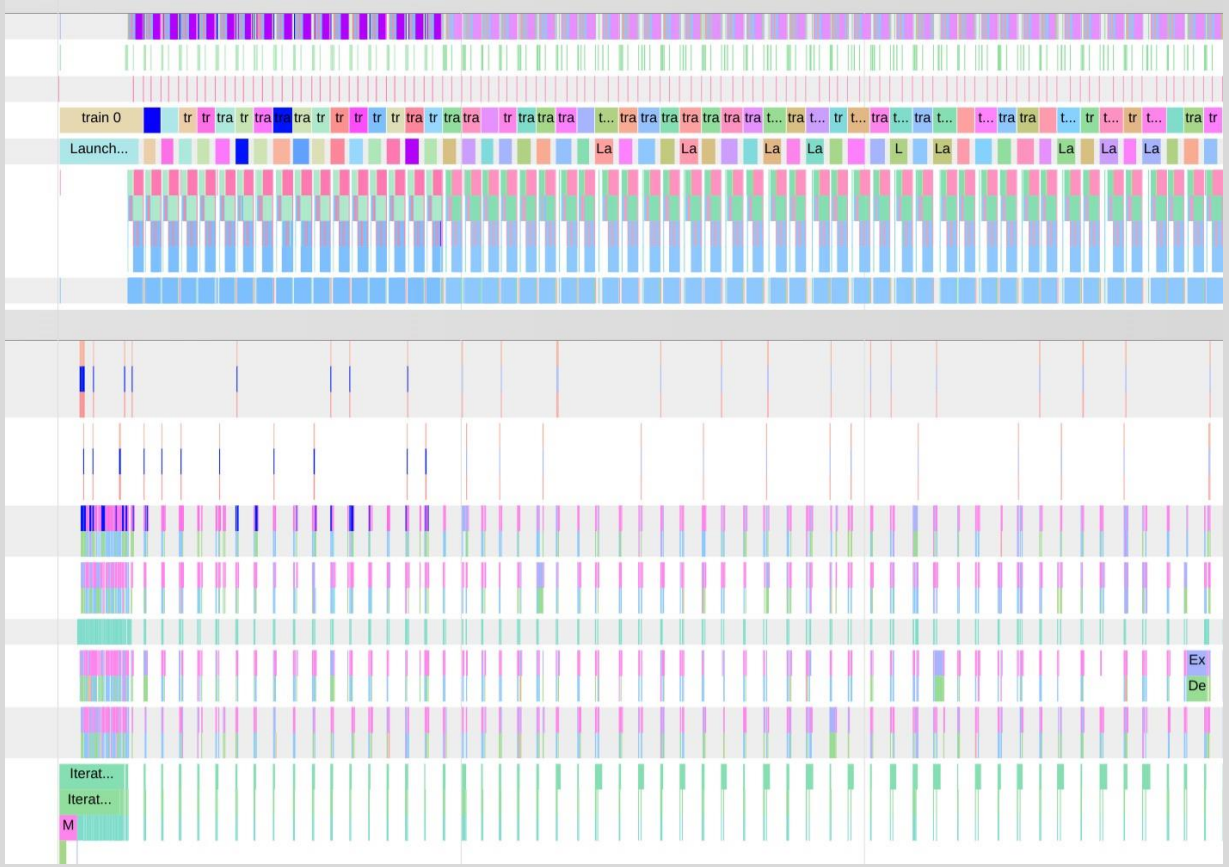
- Refer to the TF2 Profiler FAQ

Next tools to use for reducing the Device time

- tensorflow_stats (identify the time-consuming operations executed on the GPU)
- trace_viewer (look at the activities on the timeline of each GPU in the trace view)

- Says I'm not IO bound.
- About 30% of time is spent launching kernels—and what might help.
- I'm only using 32-bit operations. Trying some 16-bit may speed things up.
- It also provides links to relevant resources; which is nice.

A lot of detail from TensorFlow profiler



- A view of when operations run during the code execution (a trace).
- Hard to use this information without expert knowledge of platform implementation
- Profiling a long run will overwhelm the GUI making it unusable

Cut down on Profiling Data

- Generating profiling data for an abbreviated dataset may hide the problem being investigated.
- Instead use the same amount of data but turn on the profiler and gather data for a short time.
  - Such as one epoch (perhaps not the first).
  - Or even a set of batches within an epoch.
- This will take a snapshot of the system under load but in a usable format.
- Typically, this is done using callbacks and isn't very difficult.

Cutting Down Profiling Data

profiling_callback.py+ buffers

```
 8 class ProfilingCallback(Callback):
 9
10     def __init__(self):
11         super().__init__()
12
13     def on_epoch_begin(self, epoch, logs=None):
14         if epoch == 4:
15             tf.profiler.experimental.start('prof_dir')
16
17     def on_epoch_end(self, epoch, logs=None):
18         if epoch == 4:
19             tf.profiler.experimental.stop()
~
```

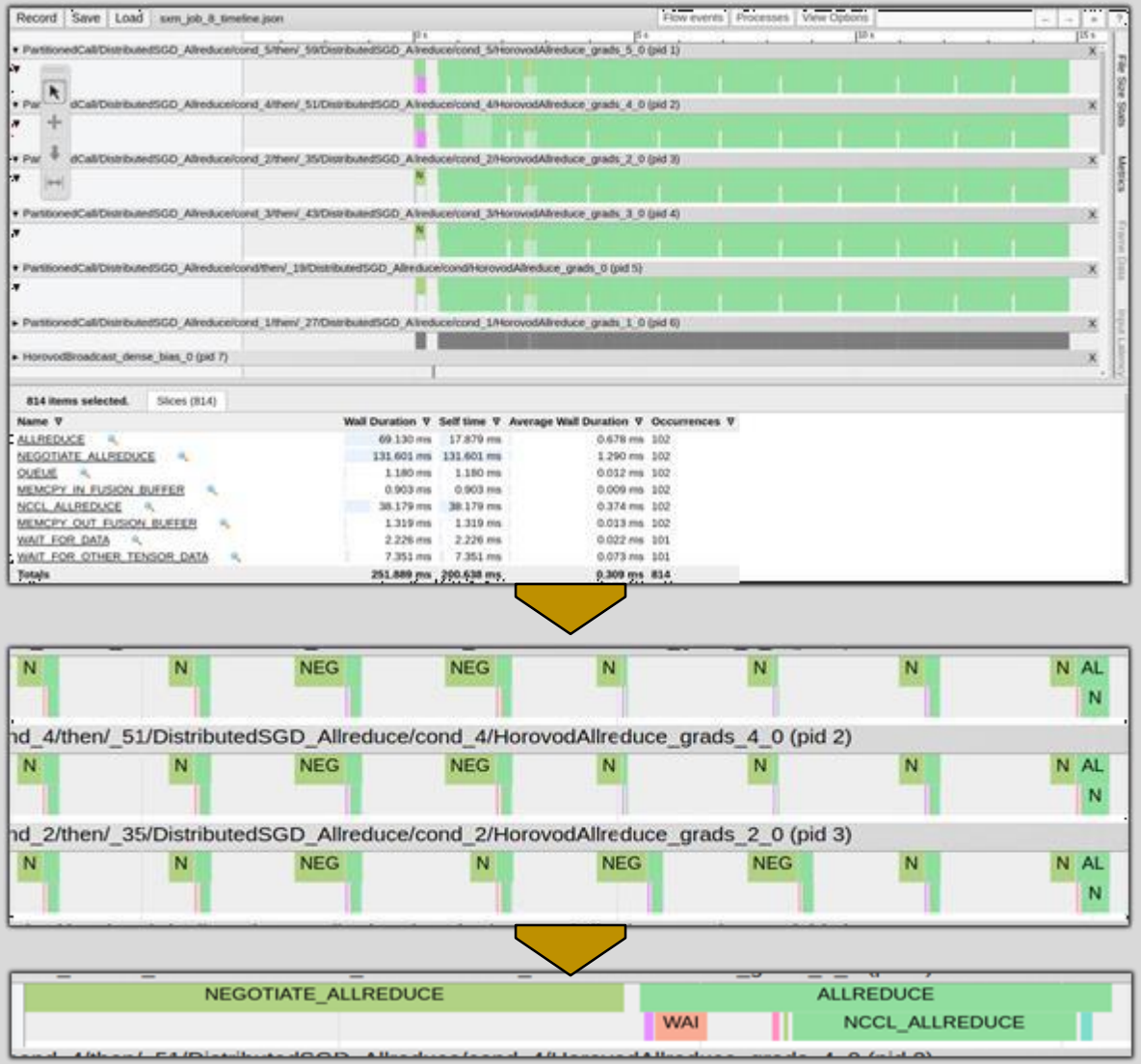
N… profiling_callback.py[+] pyt… 47% ℓN:9 %:1

- The code required to profiler the fourth epoch of a training run. There are a lot more options too. Working the system at full load but with less profiling data is very helpful.

Old Fashioned Log Entries

- A well-placed log entry can be far more useful than traces.
- The callback mechanism can also be used to generate logs.
- When running on HPC clusters logs can be written directly to files.
- Log entries can be correlated to profile data by adding and process / thread ids and timestamps.
- Entries can also be sent directly to a database in real time.
  - Jobs running on the cluster can be monitored
  - A system that runs on several nodes and GPUs can have all the data in one place.
  - Analysis is easier with a database.

Profiling multi-GPU code



- Multi-GPU frameworks (such as Horovod) have specialised profiling tools.
- These focus on inter GPU communications.
- They can tell you if too much time is being spent coordinating the work vs doing it.
- When the latest GPU technology is deployed it is sometimes necessary to use the Vendors specific profiler to augment the

Profiling multi-GPU code

- Sometimes a vender specific profile for a GPU is required.
  - NVIDIAs NCCL technology for direct memory copies aren't picked up by other profilers. Information needs to be integrated from the GPU profiler with the general ML profilers to build a real picture of what is happening.
  - This requires a lot of effort and expertise.

Profiling multi-GPU code

- Sometimes a vender specific profile for a GPU is required.
  - NVIDIAs NCCL technology for direct memory copies aren't picked up by other profilers. Information needs to be integrated from the GPU profiler with the general ML profilers to build a real picture of what is happening.
  - This requires a lot of effort and expertise.

Some Profiling Pitfalls

- Vague or incorrect representation of what is happening.
- Data can be presented in a way that obscures the underlying problem leading wasted time investigating.
- Bugs in profiling code have made the program run slow or unstable.
- Can get obsessed with performance and it can be very complex. Need to know when things are good enough.
- Sometimes a vender specific profile for a GPU is required.

Takeaway

- A mix of tools is best.
  - The ML specific profilers are very polished give good general advice.
  - Integrating that data with custom logging gives the user a good idea of what is happening.
  - Advanced tools can be added to the mix if needed but they require a lot of time and effort.
  - When running a lot of batch jobs on HPC the profiling data must be organized. Using logging features and callbacks are a good approach.
  - When there is a lot of data, a database (of some flavor) helps. Especially if data needs to be integrated.