Profiling Deep Learning

Denis Boyda, Corey Adams, Filippo Simini

2022 ALCF Computational Performance Workshop, May 24-26
Outline

• What is profiling
• Motivation to profile
• Some frequent problems and solutions
• Common Profiler APIs
• Understanding profiler output
• (new) PyTorch Profiler view
What is profiling?

Profiling is a diagnostic tool

- Measuring time/memory cost
- Identifying bottleneck
- Tracking call stack
Motivation to profile and optimization

What is the purpose of profiling?
• finding performance issues
• optimizations and tuning
• fine-tuning

Do you need to deeply profile and optimize your script?
• training vs evaluation
• Are you developing your model or you already know the solution?
• Is your accuracy increasing with scaling?

• Is speed up worth your affords?
  • Is GPU utilization high?
  • Do you expect high improvements?
  • Are you planning to scale?

“Premature optimization is the root of all evil!”
@Donald Knuth
Some optimization advices

- large kernel utilization
  - check which functions take a majority of compute

- slow IO
  - Use designated PT/TF data loaders
  - Use multithreading
  - Move preprocessing to GPU ([Nvidia DALI](https://nvidia.com/dali))

- (extra) copy CPU ↔ GPU
  - Move all operations to device
    - Rewrite your code with using only PT/TF tensors
    - Use Numba / CuPy
    - Overlap copy and computation
    - Use asynchronous copy

- low GPU utilization
  - Increase batch size

- too high precision
  - Reduce precision to float/half?
  - use autotuning mixed precision in PT
Common profiler APIs

• Context manager

```python
with profiler.profile() as prof:
    model(inputs)
```

```python
with torch.profiler.profile(
    activities=[torch.profiler.ProfilerActivity.CPU,
                torch.profiler.ProfilerActivity.CUDA],
    schedule=torch.profiler.schedule(wait=0, warmup=1, active=3),
    on_trace_ready=torch.profiler.tensorboard_trace_handler(dir_name)
) as prof:

    for _ in range(8):
        model(inputs)
```

• Decorator

```python
@profile
def train_loop(batch_size, n_training_iterations, models, opts, global_size):

    logger = logging.getLogger()

    rank = hvd.rank()
    for i in range(n_training_iterations):
```

• Python module

```bash
python -m line_profiler train_GAN.py.lprof
```

```bash
python -m torch.utils.bottleneck example2/v3.py
```
Understanding profiler output

- Additional metrics can be shown
- Reference to the line in source files can be added
- One can sort by different metrics
(new) PyTorch Profiler view

Time is measured with overlapping; the categories are ranked by priority!

1. Kernel
2. Memcpy
3. Memset
4. Runtime
5. DataLoader
6. CPU Exec
7. Other
See hands-on exercises