Node-Level Deep Learning Input Pipeline Optimization on GPGPU-Accelerated HPC Systems

28 Mar 2018

Captain Justin Fletcher
Air Force Research Laboratory
Introduction

- Deep Learning on high performance computing (HPC) systems has unique challenges
  - Shared (contested) I/O bandwidth
  - Distributed file systems
  - High compute density

- From this talk you should get:
  - The relationship between DL concurrency and I/O
  - A simple, effective method for hiding I/O on existing systems
  - An appreciation for the importance of specialized I/O systems for DL on HPC
Motivation

- Deep learning on modern HPC systems is bound by system-wide I/O
- TensorFlow optimization step running time on 4x P100, 20-core Power8 node is ~50ms/step, when:
  - System under full, mostly DL, load
  - LeNet-5 training on MNIST (batch size 256)
  - Standard asynchronous data queue
  - Typically ~16 jobs running on separate nodes
- Loading the dataset into memory yields ~17x speedup (~3ms)
- Instrumenting the training shows exhausted input queues

---

HPC is likely to remain I/O bound for the foreseeable future [1]
Impact of Queue Exhaustion

Running Time ($\mu \pm \sigma$, $n = 30$) and Queue Size ($\mu$) vs. Training Step ($I_B = 1$, Threads = 2)

Queue Exhausted $\rightarrow$ Step Running Time Increases
Causes of Queue Exhaustion: Data Parallelism

- Dequeue rate exceeds enqueue rate
- Data parallel concurrency scheme increases dequeue rate
  - Enqueue threads share storage I/O bandwidth
  - More model copies = more data throughput
  - Exacerbated by large data elements
Causes of Queue Exhaustion: Pipeline Parallelism

- Dequeue rate exceeds enqueue rate
- Model or pipeline parallel schemes increase dequeue rate
  - Typically used when model won’t fit on one device
  - Pipelines operations, increasing throughput
Standard Approach: Increase Thread Count

- Enqueue threads asynchronously enqueue data element

- Adding more enqueue threads:
  - Delays queue exhaustion
  - Decreases slowdown cause by exhaustion

- We need to increase the net enqueue rate further
  - We can’t increase enqueue rate...
  - So, we must decrease the dequeue rate.
Batch Repetition

- Artificially slow the dequeue rate by dequeuing batches less than once per step
  - Allows the queue to fill up
  - Trivial to implement

- Repeating batches introduces new problems
  - The model is optimized with less new data/step
  - Your epochs per second will decrease
  - Generalization is more impacted by how representative any individual batch is of the true data distribution
Batch Repetition Prevents Queue Exhaustion

Queue Size and Single-Step Running Time for Combinations of Batch Interval and Enqueue Threads

Distribution A. Approved for public release; distribution unlimited.
Batch Interval Impact on Net Enqueue Rate

- Running time is inversely proportional to net enqueue rate
  - Validates the hypothesis that training was I/O bound
  - We get diminishing returns for batch intervals >16
- Batch intervals allow for better throughput with less threads

Distribution A. Approved for public release; distribution unlimited.
Batch Interval & Model Performance

- Validation Loss at Convergence (Step 10000)
- MNIST Images per Second
- Batch Interval values: 1, 2, 4, 8, 16, 32, 64

Validation Loss decreases as the batch interval increases, indicating improved model performance at lower intervals.
Summary

- HPC systems are structurally likely to be I/O bound for DL workloads
- Repeating batches for an interval of steps can hide I/O latency and keep the GPUs fed
- Small refresh intervals don’t impact converged optimization, but decrease runtime
- If you want to talk more, ask me about my circular data queues
Distribution A. Approved for public release; distribution unlimited.
Global Exemplar Architectures
Relative to Fastest HPCMP System

- I/O bandwidth (terabytes/s)
- Disk capacity (petabytes)
- 1/(interconnect latency) (1/milliseconds)
- Interconnect bisection BW (petabytes/s)
- Memory capacity (petabytes)
- Job duration (hours)
- PetaFLOP/s

HPCMP
Onyx
(2017)
China
TaihuLight
(2016)
ORNL
Summit
(2018)
Exascale
Reference
(2023)