LBANN: Livermore Big Artificial Neural Network Toolkit,
ONNX: Open Neural Network Exchange

AU21 CSE 5449
Yuting Fang.564
Outline

LBANN: Livermore Big Artificial Neural Network Toolkit
  • Introduction: HPC, Scalability
  • Implementation: MPI + Thread
  • Basic Usage: Protobuf and Python Frontend
  • Parallelism Strategy: model + data, fine grained data parallelism

ONNX: Open Neural Network Exchange
  • Introduction
  • Example
Introduction: LBANN

• Frist Published: 2015

• Livermore Big Artificial Neural Network: an HPC-centric deep learning training framework that works across multiple levels of parallelism.

• Modern HPC characteristics:
  • Low latency interconnect
  • High power CPUs
  • Node-local storage
  • Tightly-coupled GPUs
  • Highly-optimized communication libraries

Scalability of Deep Learning

• “This paper presents our preliminary results in scaling the size of model that can be trained with the LBANN toolkit.”

• **Strong Scaling**: solve a fixed sized model faster
  ▪ Same global minibatch size, more GPUs $\rightarrow$ smaller per-node minibatch size
  ▪ Reaches a point where communication overhead is too high

• **Weak Scaling**: solve a larger problem in fixed time
  ▪ Same per-node minibatch size, more GPUs $\rightarrow$ larger global minibatch size
  ▪ May not being faster
Implementation: MPI + Threads

**MPI: Message Passing Interface**
- Enable communication between processes, optimize data movement
- Parallelism between processes (across nodes)
- Exploiting distributed processing and communication

**Thread**
- MPI processes have multiple threads
- Provide a shared-memory model within a process (within a node)
- Exploiting node-local thread-level parallelism through BLAS library
Scalable Deep Learning Software Stack

- LBANN
  - Scalable Deep Learning Toolkit
- Hydrogen
  - GPU-Accelerated
  - Distributed Linear Algebra
- Aluminum
  - High-performance GPU-aware communication library

- Optimized distributed memory algorithms
- Pythonic “PyTorch-based” model description
- Support for ONNX model exchange

- MPI
- CUDA-aware MPI
- NCCL + custom
- NVSHMEM

- C++ / MPI + OpenMP / CUDA
- Open-sourced on github.com
Basic Usage: Frontends

1. Protobuf Frontend
   • Protocol buffers: a language-neutral, platform-neutral extensible mechanism for serializing structured data developed by Google.
     -- implement Remote Procedure Call, widely used to support distributed services
   • The main LBANN driver uses Protobuf text files to specify experiments.
   • “Aside from quick debugging, there are very few situations where directly manipulating Protobuf text files is superior to using the Python frontend.”

2. Python Frontend (PFE)
   • A convenience wrapper around the Protobuf frontend.
   • Syntax similar to Pytorch

Question

LBANN is Define-and-Run.
Does Define-and-Run fit HPC better than Define-by-Run?

Multiple-Level Parallelism Strategy

Challenges for *large* model and data sets:

1. Massive dataset
   -- large number of samples
2. Large sample size
   -- large single sample, need large model to fit
3. Large network
   -- large number of network parameters

• Why data parallelism is *not enough* here?
Data Parallelism

• Samples are distributed across processors.
• Most Common.

• Scalability Limitation:
  • **Limited memory capacity**
    - Neural networks are becoming deeper, do not fit GPU memory
  • **Limited Parallelism**
    - Degree of parallelism depends on minibatch size $N$
    - Can’t make $N$ arbitrary large, as accuracy drops
LBANN Parallelism Strategy

Challenges for large model and data sets:
1. Massive dataset
2. Large sample size
3. Large network

➤ designed for robust **model parallelism** with modest **data parallelism**
• Model Parallelism: distribute model across processors
Parallelism Strategy in LBANN

Figure 1: LBANN Distributed neural network model with parallel fetch from Lustre PFS

Figure 2: Distributing mini-batches

(a) Processing mini-batch 0

(b) Processing mini-batch 1
Evaluation: Train an Autoencoder

• Test: 50k neurons models (9.8 billion parameters, 73GB), 8-128 nodes, mini-batch sizes from 8-2048 images.

• Large mini-batches benefit greatly from additional nodes.

• Smaller mini-batches limited improvement: dominated by communication overheads.

Figure 5: Strong scaling 50K neuron hidden layer: per mini-batch training time vs. number of nodes.

Exploiting Model Parallelism
LBANN Parallelism Strategy

Challenges for large model and data sets:

1. Massive dataset
2. Large sample size
3. Large network

• designed for robust model parallelism with modest data parallelism
• provides model-parallel acceleration through domain decomposition to optimize for strong scaling of network training

Generalized Parallel Convolution in LBANN

Sample (Data) Parallelism

Spatial Parallelism

Channel/Filter Parallelism

Allreduce

Halo exchange

Allgather/ReduceScatter

O(100-1000) GPUs

O(10) GPUs

O(10) GPUs

NCCL/MPI

MPI/Custom

NCCL/MPI

IPDPS’19

SC’19

Allreduce

Halo exchange

Allgather/ReduceScatter

O(10) GPUs

O(10) GPUs

NCCL/MPI

MPI/Custom

NCCL/MPI
Expressing Parallel Convolution

Example: 2D image

- **Input** - \( x: N \times C \times H \times W \)
- **Filters/Weights** - \( w: F \times C \times K \times K \)
- **Output** - \( y: N \times F \times H \times W \)
- **Gradients**: \( \frac{dL}{x}, \frac{dL}{w}, \frac{dL}{y} \)

- Use * to represent a replicated dimension
- Use ° to represent a distributed dimension
- Data parallelism – \( x[^\circ, *, *, *, *] \)
Spatial Parallelism: $\mathbf{x}[*, *, °, °]$

- Distributing individual samples across height and width
- Halo Exchange: provide access to remote data
- Fits larger spatial domains.

Near perfect weak scaling, excellent strong scaling.
Introduction: ONNX

• Initial release: 2017

• Open Neural Network Exchange: “An open ecosystem that empowers AI developers to choose the right tools as their project evolves. Provides an open source format for AI models.”

  focusing on the capabilities needed for inferencing

• Although one framework may be best for one stage of a project’s development, another stage may require a different framework.

Reference: https://github.com/onnx/onnx
Reality

Training Framework

Deployment Target

Intermediate Representation in DL

• Deep learning with neural networks is accomplished through computation over dataflow graphs.

• These graphs serve as an Intermediate Representation (IR) that
  ■ capture the specific intent of the developer's source code
  ■ conducive for optimization and translation to run on specific devices

• ONNX format: a common IR

• Specification
  ■ Definition of an extensible computation graph model
  ■ Definitions of standard data types
  ■ Definitions of built-in operators

Reference: https://github.com/onnx/onnx/blob/master/docs/Overview.md
Converters for popular frameworks

• An intermediary machine learning framework used to convert between different machine learning frameworks
  -- Chainer: chainer/chainer-onnx (not support dynamic models)
  -- Tensorflow: onnx/tensorflow-onnx, onnx/onnx-tensorflow
  -- PyTorch (native export)
  -- Scikit-learn: onnx/sklearn-onnx
  -- CoreML: onnx/onnx-coreML
  -- ...

• Tutorials
Sources

• LBANN: livermore big artificial neural network HPC toolkit, https://dl.acm.org/citation.cfm?id=2834897
• Nikoli Dryden et al., Improving Strong-Scaling of CNN Training by Exploiting Finer-Grained Parallelism (IPDPS ‘19), https://arxiv.org/abs/1903.06681
• Naoya Maruyama, LLNL, Scalable Distributed Training of Large Neural Networks with LBANN, https://www.youtube.com/watch?v=P2UIQxYw900
• Brien van E, LLNL, Deep Learning at Scale for Scientific Computing, https://www.youtube.com/watch?v=VXBulbL_iDg
• ONNX, https://github.com/onnx