Building a Scalable Deep Learning Platform

Sayed Hadi Hashemi
Benjamin Rabe
Volodymyr Kindratenko
NFS-funded Project

- NSF Program: Major Research Instrumentation (MRI)
  - Award Number: 1725729
  - Start Date: October 1, 2017
  - NSF Funding: 70% of the total budget
    - Hardware
    - One postdoc
    - One systems engineer/research programmer
- UofI cost-share: 30%
  - 0.25 FTE of a co-PI
  - One postdoc
  - 2 GRAs
  - Some hardware
Project Objectives

- Develop and deploy a novel instrument for accelerating deep learning research at the University of Illinois
  - The instrument will integrate the latest computing, storage, and interconnect technologies in a purpose-built shared-use system
  - It will deliver unprecedented performance levels for data-intensive applications that use on deep leaning
- The instrument development is
  - driven by the UofI deep learning community needs and
  - carried out in collaboration with IBM and NVIDIA
- The instrument will serve as a focal point for the rapidly growing deep learning research community at UofI, enable expansion of several research programs, and contribute to STEM education and training
Early System

- IBM 8335-GTH AC922 server (8335-GTG early shipment program)
  - Dual-socket server
    - 2x 20-core 2.0 GHz [2.87 GHz Turbo]
    - SMT4
  - 8 memory channels per socket
  - 120 GB/s memory bandwidth per socket
  - 512 GB DDR4 RAM
- 2x 3.84 TB Disk Drives
- 4x NVIDIA V100 GPUs
- NVLink 2.0
- 1.6TB SSD NVMe PCIe drive
- CAPI-enabled FPGA board [later]
IBM 8335-GTG AC922 vs NVIDIA DGX-1 P100

- Early results with TensorFlow 1.5.0

DGX-1 Results from https://www.tensorflow.org/guide/performance/benchmarks
Where we are now

- Resnet-50 v1, FP16, Batch Size=1024, DAWNBench

Workloads

- **Deep Learning Training**
  - +90% of jobs uses one node only
  - Goal is to reduce the training time
  - Scaling up is more important than scaling out

- **Interactive Training**
  - One GPU via Jupyter Notebooks
  - The goal is faster availability

- **Inference**
  - The goal can be either low latency or high throughput
  - Example: Deep Reinforcement Learning, Live Feed

- **Data Science**
  - Non Deep Learning Workloads
  - Pandas, numpy, ...
Deep Learning Training

- Find a general pattern on seen examples to predict unseens in future!

- It a optimization problem in heart:
  - Data
    - Example: $x$
    - Desired Prediction [Label]: $y$
  - Model
    - Predictor: $\phi(x)$
    - Parameters: $w$
  - Goal:
    - $\min_w \sum [\phi(w,x) - y]^2$
Stochastic Gradient Descent & co

\[ \mathbf{w} \leftarrow \mathbf{w}_0 \]

Repeat until an approximate minimum is obtained:

Randomly shuffle examples

\[ \mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \sum (\phi(\mathbf{w}, \mathbf{x}) - y)^2 \]

Parameters \quad \downarrow \quad \text{ML Model} \quad \downarrow \quad \text{Loss Function} \quad \downarrow \quad \text{Input Data} \quad \downarrow \quad \text{Gradients} \quad \downarrow \quad \text{Learning Rate}
$w \leftarrow \text{Parameter}$

$\phi \leftarrow \text{Input Data}$

$x, y \leftarrow \text{Input Data}$
Parameter

Read

Update

Input Data

Read

Calculating Loss

Input Data

Calculate Gradients

Forward Pass

Back propagation

Update

Iteration

11
Parameter

Input Data

Update

Iteration

Read

Input

Forward Pass

Back propagation

Update
Performance

Accumulivity

Deep Learning

Traditional ML

Data + Model Size

Idea

Results

Code
Example: ImageNet Challenge

- **Object Detection on ImageNet Challenge**
  - # images = 1,281,167
  - # Epoch = 30

- **CPU Performance** (Power 9, 40 cores)
  - $1,281,167 \times 30 / 4.7 = 93$ days 7 hours 3 minutes

- **GPU Performance** (V100):  
  - $1,281,167 \times 30 / 36 = 12$ days 8 hours 33 minutes

*Image Credit: ImageNet*
Performance

- **Goal:** to reduce the training time to the certain accuracy:  
  \[ \text{Average Step Time} \times \text{Number of step} \]

- Decrease the Step Time (System Challenge)
- Decrease the number of steps (Machine Learning Challenge)
- Often these two goals have inverse correlation!
  - e.g Asynchronous Communication
Scaling Up

• Successful scaling needs domain knowledge of both systems and machine learning

• Optimizations:
  • Minibatch Size
  • Floating-point Precision
  • Input Pipeline
  • Dynamic Input Size
  • Multi Device Training
Minibatch Size

Accelerators are SIMD/SIMP

Larger mini batch size results in better throughput

But it needs careful hyper-parameter tuning
  - And more [1]
  - Sometime it does not work at all

Example:
  - Resnet-50 v1
  - **CPU**: Power 9, 40 CPU cores
  - **GPU**: Power 9, V100 GPU

Floating Point Precision

- FP precision reduces the step time
  - But increases the number of steps
  - Needs large mini batch size

- Using FP16 needs additional consideration:
  - FP32 storage
  - Loss Scaling
  - And more [2]

- Example:
  - Resnet-50 v1
  - bs=128
  - GPU: V100

**Input Data Pipeline**

- **Input Data Example:** Resnet
  - Read Multiple Random Small Images
  - Resize
  - Normalize
  - Add Noise
  - Batch Together

- **Optimization**
  - Pseudo Random Shuffling
  - Store in Special Format (HDF5, TFRecord)
  - Parallel Data Processing [3]
  - Prefetch multiple steps
  - Currently 23 Datasets

- **Example:**
  - Resnet-50 v1
  - bs=128
  - Parallel Data Processing = 4
  - Prefetch Steps = 16
  - GPU: V100

---

[3] https://www.tensorflow.org/api_docs/python/tf/data
Dynamic Input Size

- Smaller input size reduces the step time
- But increases the number of steps
- Start with small size and gradually increase the size
- And more [4]

Example:
- ResNet-50 v1
- Default size: 224x224
- bs=128
- GPU: V100

Distribution

- Data Parallel
  - Each device has a copy of the model
  - Divide the batch size among multiple devices
  - Aggregate gradients before applying

- Data Replica
  - Divide the model among multiple devices
  - Wire up the devices
Parameter Server

All Reduce

- Resnet-50 v1, bs=32
- BlueWaters
- All Reduce: Ring
Collective Transfers
Collective Transfers
Collective Transfers
Multi Device

- Resnet-50 v1, bs=128 per device
- Need hyper parameter adjustment

Credit: Facebook Research
Distribution

![Bar chart showing comparison of distribution with Ours, Horovod, and DDL. The chart compares performance at 4 Gpus / 1 Node and 8 Gpus / 2 Nodes.]
Next Steps

• When Data Parallel Fails:
  • Sometime minibatchsize can not be increased:
    • Number of steps increases
    • GPU Memory limits the batch Size
  • Model Parallel
    • Tensor2Tensor
    • Mesh TensorFlow [5]

• Embarrassingly Parallel Jobs:
  • Hyperparameter Tuning
  • AutoML

Isolation

- Containers
- Provide flexibility and isolation
- Low overhead

- Example:
  - Resnet-50 v1
  - bs=128
  - GPU: V100
Next steps: 16-node cluster

- BMC Network (1Gb)
- Compute & Storage Network (EDR IB)
- Management and Access Network (1Gb)
- BMC/Mgmt Node
- Storage System (TBD)
- Compute Node 1
- Compute Node 2
- Compute Node 16
- Test & Devel. Node
- Data Node
- Login Node

Public Network
Next Steps: Inference on FPGA

- Nallatech 250S+ CAPI FPGA board
- 8-lane PCI-Express Gen 4
- Xilinx Kintex Ultrascale+ FFVA1156
- 4GB DDR4 SDRAM, 2400 MT/s
- 4x 960GB NVMe SSD sticks
Questions?