Outline

- Motivation
- Poseidon Implementation
- Results
- Future work & Discussion
- Quick Demo
Motivation
Deep Learning is time consuming

Qualitative Changes
- Larger dataset
- Deeper networks
- Less time
- Less Money
Goal

• Scale DL training to distributed GPU cluster
BackPropagation\text{(Mini\text{-}batch)}

\[ A^{(t)} = A^{(t-1)} + \epsilon \cdot \sum_i \nabla(A^{(t-1)}, D_i) \]

- A: weights
- Di: training sample i
BackPropagation(Mini-batch)

\[ A^{(t+1)} = A^{(t)} + \epsilon \sum_{p=1}^{P} \nabla_{\ell}(A^{(t)}, D^{(t)}_p) \]

- **A**: weights
- **P**: workers
- **D_p**: batch of training samples in worker p
Training Iteration

$$[C_t, S_t] = [f_t^1, \cdots, f_t^L, b_t^L, \cdots, b_t^1, s_t^L, \cdots, s_t^1]$$

$$s_t^l = [o_t^l, i_t^l]$$

- $C_t$: computation process in $t$
  - $f_t + b_t$ (forward + backward)
- $S_t$: synchronization process in $t$
  - $o_t + i_t$ (push + pull)
Poseidon Idea I

- Wait-free Backprop (WFBP)
- Layer by layer backprop
- Overlap Ct and St
- Improve GPU utilities
AlexNet (61.5M parameters)  
Titan X GPUs  
Batch size 256  
240Mi/s  

Cost: 240Mi * sizeof(float)  
= 840 MB/s  
> Commercial Ethernet Bandwidth
Poseidon Idea II

- Hybrid Communication
  - Honer PS
  - Honer SFB
- Self-estimated for different layers
**Sufficient Factor Broadcasting**

- Communication Cost
  - PS: $2 \times \text{#workers} \times w \times h$ (optimal for big cluster size and small $W$)
  - SFB: $2 \times k \times (w + h) \times \text{#workers} \times \text{#workers}$ (optimal for large matrix $W$ between FCLs)

- $k$: batch size, $w$: #row, $h$: #column
Implementation
Algorithm 1 Get the best communication method of layer $l$

1: function METHOD($layer$ $l$)
2:     $layer$.property = Query($l$, 'property')
3:     $n_1$, $n_2$, $k$ = Query('n_worker', 'n_server', 'batchsize')
4:     if $layer$.property.type != 'FC' then
5:         return 'PS'
6:     else
7:         $w = layer$.property.width
8:         $h = layer$.property.height
9:         if $2k(n_1 - 1)(w + h) \leq 4wh(n_2 - 1)/n_2$ then
10:            return 'SFB'
11:        else
12:            return 'PS'
13:        end if
14:    end if
15: end function

Algorithm 2 Parallel training at worker $p$

1: function TRAIN($net$)
2:     for $iter = 1 \rightarrow T$ do
3:         $sync\_count = 0$
4:         $net$.Forward()
5:         for $l = L \rightarrow 1$ do
6:             $net$.BackwardThrough($l$)
7:             job.queue.Push.back($l$)
8:             if $thread\_pool$.size() < max_thread then
9:                 thread_pool.Push.back(&$sync\_job$)
10:         end if
11:     end for
12:     while $sync\_count < net.num\_trainable$ do
13:         wait()
14:     end while
15: end function

function SYNC\_JOB()

16: stream = stream_pool.Allocate()
17: while 1 do
18:     $l = job\_queue$.Wait\_and\_pop()
19:     syncers[$l$].Move(stream, GPU2CPU)
20:     syncers[$l$].method = coordinator.Best\_Method($l$)
21:     syncers[$l$].Send()
22:     syncers[$l$].iter++
23:     syncers[$l$].Receive()
24:     syncers[$l$].Move(stream, CPU2GPU)
25:     $sync\_count++$
26: end while
27: end function
Results
Scalability

Figure 4. Throughput scaling when training (a) GoogLeNet (b) VGG19 (c) VGG19-22K using Poseidon-parallelized Caffe and 40GbE bandwidth. The throughput of original Caffe on a single GPU is set as baseline (i.e. speedup = 1).
Scalability (cont’d)

Figure 5. Throughput scaling when training (a) Inception-V3 (b) VGG19 (c) VGG19-22K using Poseidon-parallelized TensorFlow and 40GbE bandwidth. The throughput of original TensorFlow on a single GPU is set as baseline.
Figure 6. Throughput scaling when training (a) GoogLeNet (b) VGG19 (c) VGG19-22K using Poseidon-parallelized Caffe with varying network bandwidth. The throughput of original Caffe on a single GPU is set as baseline (speedup = 1).
Bandwidth (cont’d)

**Figure 7.** Breakdown of computation and stall time of GPUs when training Inception-V3, VGG19 and VGG19-22K on 8 machines using different systems.
Future Work & Discussion
• Discussion

• How to apply Poseidon to Chain-like nets?

• How to reduce communication time further for low bandwidth, network congestion?
  • More communication strategies for different layers

• How to estimate communication cost for dynamic nets?
  • Simple high-level interface
  • ML classifier to dispatch

• How does industry companies manage GPU clusters?
  • Shared Cluster environment(docker bind)
  • SAAS users
Future Work

- New backend design and implement
  - Dynamic nets (e.g. dynet) integration
  - General purpose ML models
  - Optimize unsupervised, reinforcement learning models
- Distributed logic refactor and extend
- Cluster management
  - GPU, Memory
- Other devices
Conclusion

• Scalable efficient communication library for distributed DL

• Easy to plug in(impl) Caffe, TensorFlow, MxNet...

• Reference: Hao and Zeyu’s paper!

• Try it out at http://poseidon-release.readthedoc
Release Status

- Implemented in TensorFlow 0.10
- Similar develop interface with tf
- Refined Keras Interface (TODO)
- Deployment ready
  - wheels
  - AMIs
  - rpm, deb packages (TODO)
Quick Demo
Thanks