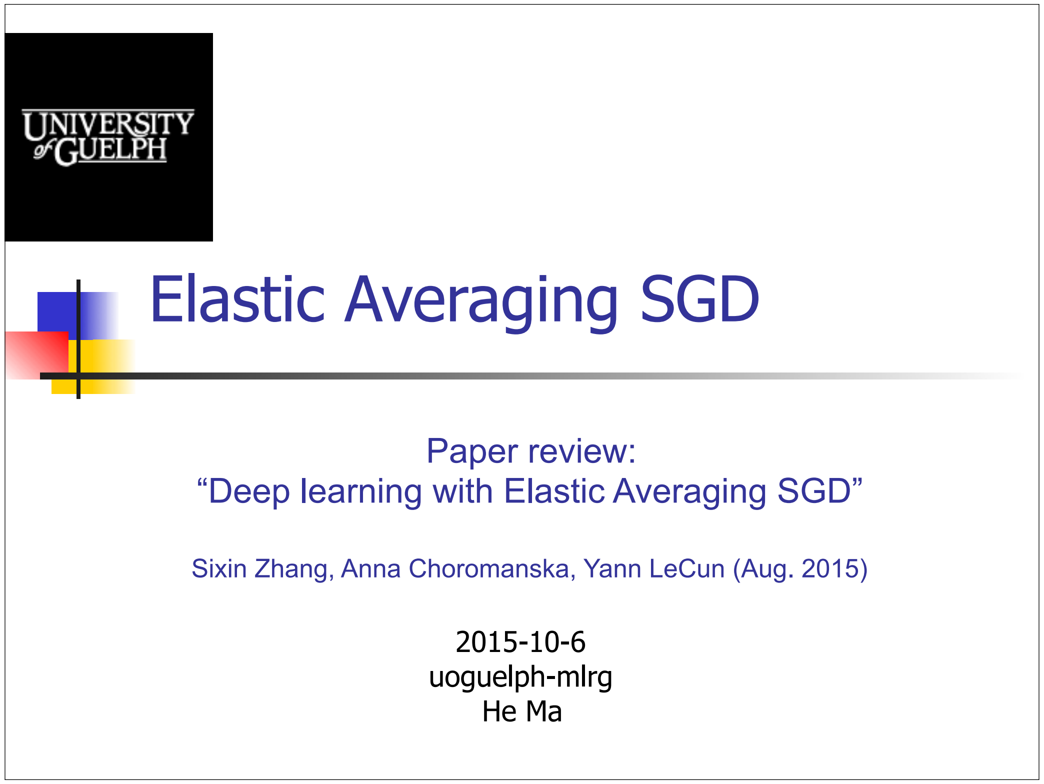


Elastic Averaging SGD

Paper review:
"Deep learning with Elastic Averaging SGD"

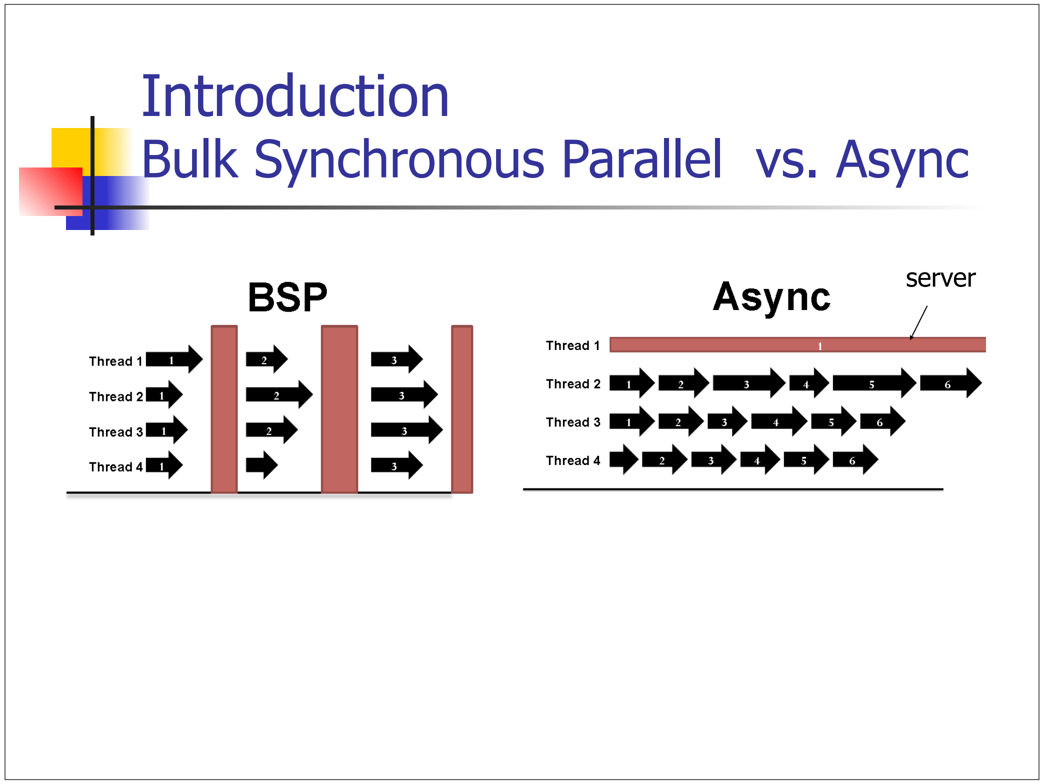
Sixin Zhang, Anna Choromanska, Yann LeCun (Aug. 2015)

2015-10-6
uoguelph-mlrg
He Ma

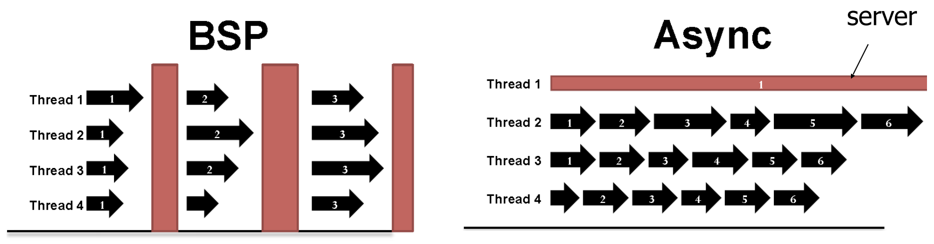
Sixin is a Phd student from NYU.
Anna is a Post-Doctoral Associate in NYU.
Yann LeCun is the well known professor in NYU and research scientist from facebook. He contributed greatly in the field of character recognition and is the founding farther of Convolution Neural Networks.
Accepted on Cornell university archive

Outline

- Introduction (why parameter server, what does it look like)
- Related Works (BSP,Async Parallel, SSP)
- Main contribution (EASGD and Proof)
- Testing (Setup and results)
- Summary

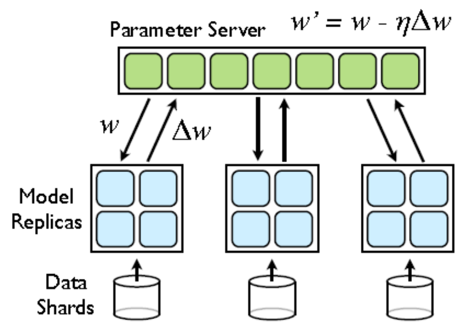


Introduction
Bulk Synchronous Parallel vs. Async



Those pictures are from the petuum slides.
BSP is the structure our current implementation is using.
Async is the Downpour structure.

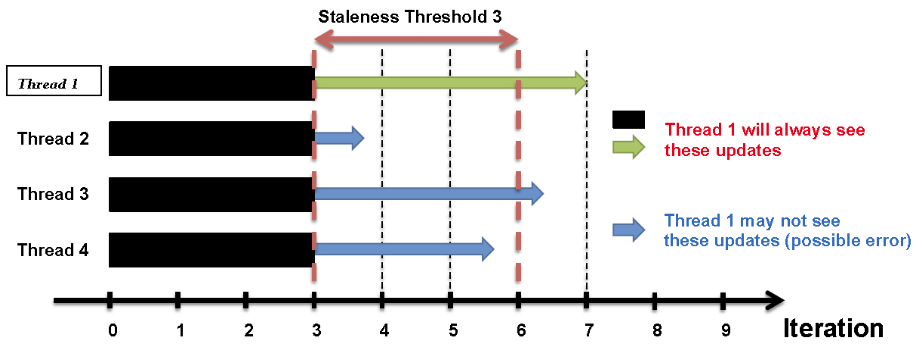
Introduction
Downpour structure (Async)



Credit: “Large Scale Distributed Deep Networks ” 2012 Jeffrey Dean et al.

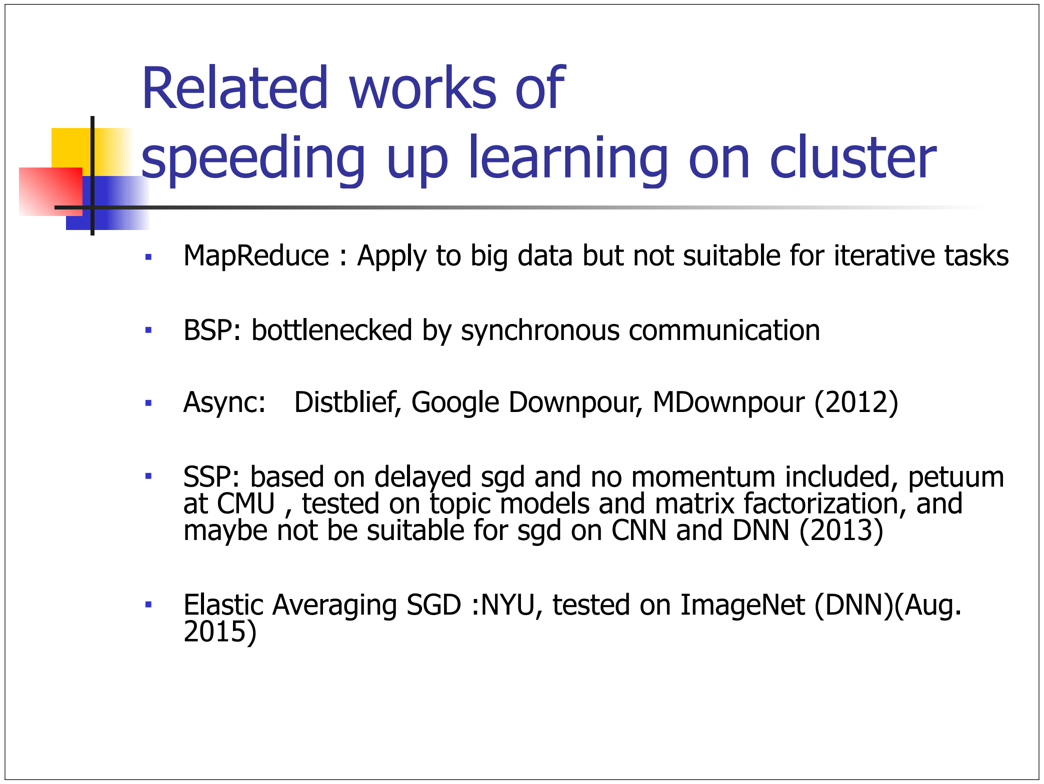
Here it what a parameter server looks like.
So there can be another problem that is the server bandwidth wide enough to accommodate the number workers.
And staleness exists. That is to say a work can fetch a very old version of W.

Introduction
Stale Synchronous Parallel(SSP)



Staleness Threshold 3
Thread 1
Thread 2
Thread 3
Thread 4
Thread 1 will always see these updates
Thread 1 may not see these updates (possible error)
0
1
2
3
4
5
6
7
8
9
Iteration

In 2013, the researchers from Carnegie mellon university.
In their implementation, they define the staleness of a parameter updating scheme.
Control the staleness so that workers will not fetch too old W.
The faster worker will wait for the slowest work when the slowest worker fall too far behind.



Related works of speeding up learning on cluster

- MapReduce : Apply to big data but not suitable for iterative tasks
- BSP: bottlenecked by synchronous communication
- Async: Distblief, Google Downpour, MDownpour (2012)
- SSP: based on delayed sgd and no momentum included, petuum at CMU , tested on topic models and matrix factorization, and maybe not be suitable for sgd on CNN and DNN (2013)
- Elastic Averaging SGD :NYU, tested on ImageNet (DNN)(Aug. 2015)

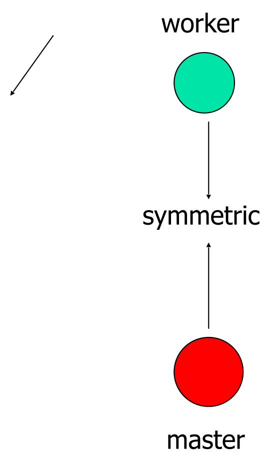
Our implementation of distributed Deep Learning is using MPI or P2P socket comm, the comm overhead is quite high when using larger than 8 ranks. So we need to either improve the communication time or pursue better substite of async algorithm.

Elastic Averaging SGD

- EASGD update rule:

worker $- \tilde{x}^{t}))$

master



The main contribution of the paper is the new algorithm “EASGD” .
t is the iteration number the worker is currently in.
Master side has a separate iteration counter.
Whenever the master and a worker part, the rule will constraint them towards each other.

Elastic Averaging SGD cont.

Credit: http://www.usatoday.com/story/news/world/2015/03/18/berlin-dog-leash-law/24956997/

This may not be a good analogy. But it is very like you have some leashed hounds in search of something. Every hounds may go in different direction in the solution space. They will drag the master around.

Elastic Averaging SGD cont.

- EASGD update rule: let

Two parts: data driven and leashed by master

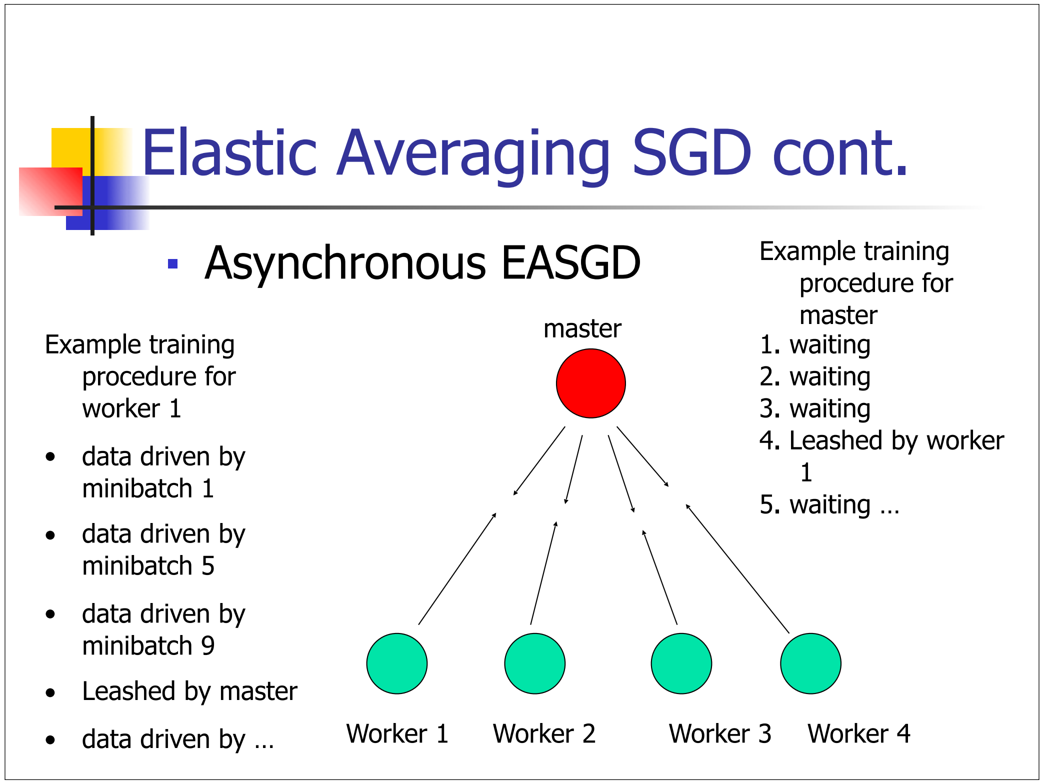
$- \tilde{x}^t)$

$\sum_{i=1}^{p} x^t_i)$

$-$

Leashed by many workers

The “averaging” comes from here, this term 1/p sum(x_t).

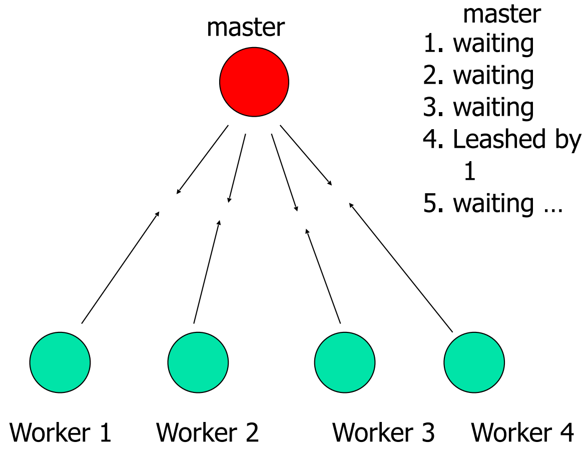


Elastic Averaging SGD cont.

- Asynchronous EASGD

Example training procedure for worker 1

- data driven by minibatch 1
- data driven by minibatch 5
- data driven by minibatch 9
- Leashed by master
- data driven by …

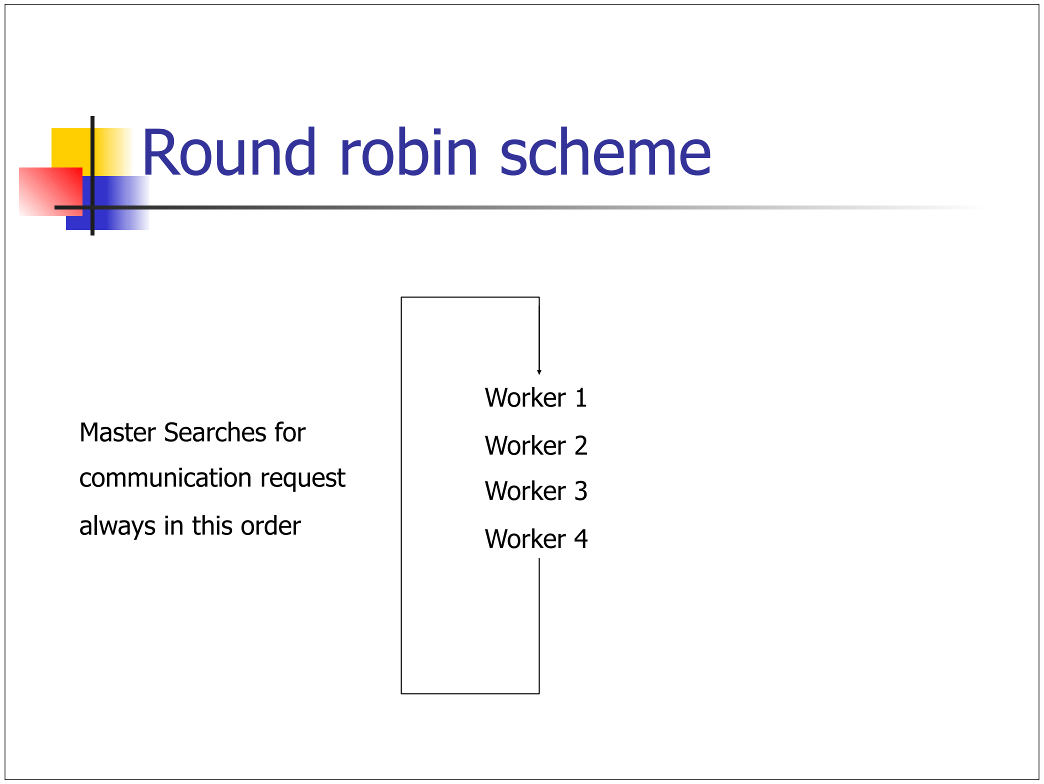


Example training procedure for master

1. waiting
2. waiting
3. waiting
4. Leashed by worker 1
5. waiting …

Considering that synchronous version of the algorithm, the master need to communicate with every worker. If the bandwidth at the master side is not wide enough. This gather and broadcast operation can be slow. And all workers have to work at the same pace even though someone maybe slower than others.

In the paper, they proposed an asynchronous version of the EASGD. The master sevices the workers in a round robin (one by one) fasion.



Round robin scheme

Master Searches for communication request always in this order

Worker 1

Worker 2

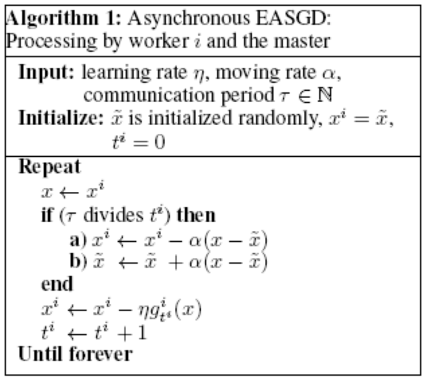
Worker 3

Worker 4

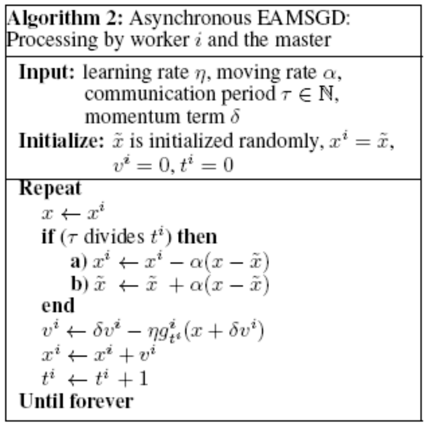
To prevent starvation of any worker.

Elastic Averaging SGD cont.

- Asynchronous EASGD with momentum



```
Algorithm 1: Asynchronous EASGD:
Processing by worker $i$ and the master
Input: learning rate $\eta$, moving rate $\alpha$,
       communication period $\tau \in \mathbb{N}$
Initialize: $\tilde{x}$ is initialized randomly, $x^i = \tilde{x}$,
       $t^i = 0$
Repeat
    $x \leftarrow x^i$
    if ($\tau$ divides $t^i$) then
        a) $x^i \leftarrow x^i - \alpha(x - \tilde{x})$
        b) $\tilde{x} \leftarrow \tilde{x} + \alpha(x - \tilde{x})$
    end
    $x^i \leftarrow x^i - \eta g^i_{t^i}(x)$
    $t^i \leftarrow t^i + 1$
Until forever
```



```
Algorithm 2: Asynchronous EAMSGD:
Processing by worker $i$ and the master
Input: learning rate $\eta$, moving rate $\alpha$,
       communication period $\tau \in \mathbb{N}$,
       momentum term $\delta$
Initialize: $\tilde{x}$ is initialized randomly, $x^i = \tilde{x}$,
       $v^i = 0, t^i = 0$
Repeat
    $x \leftarrow x^i$
    if ($\tau$ divides $t^i$) then
        a) $x^i \leftarrow x^i - \alpha(x - \tilde{x})$
        b) $\tilde{x} \leftarrow \tilde{x} + \alpha(x - \tilde{x})$
    end
    $v^i \leftarrow \delta v^i - \eta g^i_{t^i}(x + \delta v^i)$
    $x^i \leftarrow x^i + v^i$
    $t^i \leftarrow t^i + 1$
Until forever
```

The leash or elastic part is the same.
Tau is the communication period. Larger tau means that less frequent communication but have the risk of workers drifting too far and a strong leash force when this is found.
The data driven part is replaced with velocity.

Proof of EASGD stablility

- Composite map

ADMM

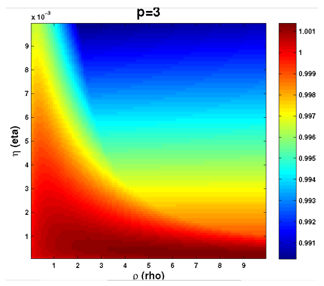
$$\lambda^i_{t+1} = \begin{cases} \lambda^i_t - (x^i_t - \tilde{x}_t) & \text{if} \quad \mathrm{mod}\,(t,p) = i-1; \\ \lambda^i_t & \text{if} \quad \mathrm{mod}\,(t,p) \neq i-1. \end{cases}$$

$$x^i_{t+1} = \begin{cases} \frac{x^i_t - \eta \nabla F(x^i_t) + \eta\rho(\lambda^i_{t+1} + \tilde{x}_t)}{1+\eta\rho} & \text{if} \quad \mathrm{mod}\,(t,p) = i-1; \\ x^i_t & \text{if} \quad \mathrm{mod}\,(t,p) \neq i-1. \end{cases}$$

$$\tilde{x}_{t+1} = \frac{1}{p}\sum_{i=1}^{p}(x^i_{t+1} - \lambda^i_{t+1}).$$

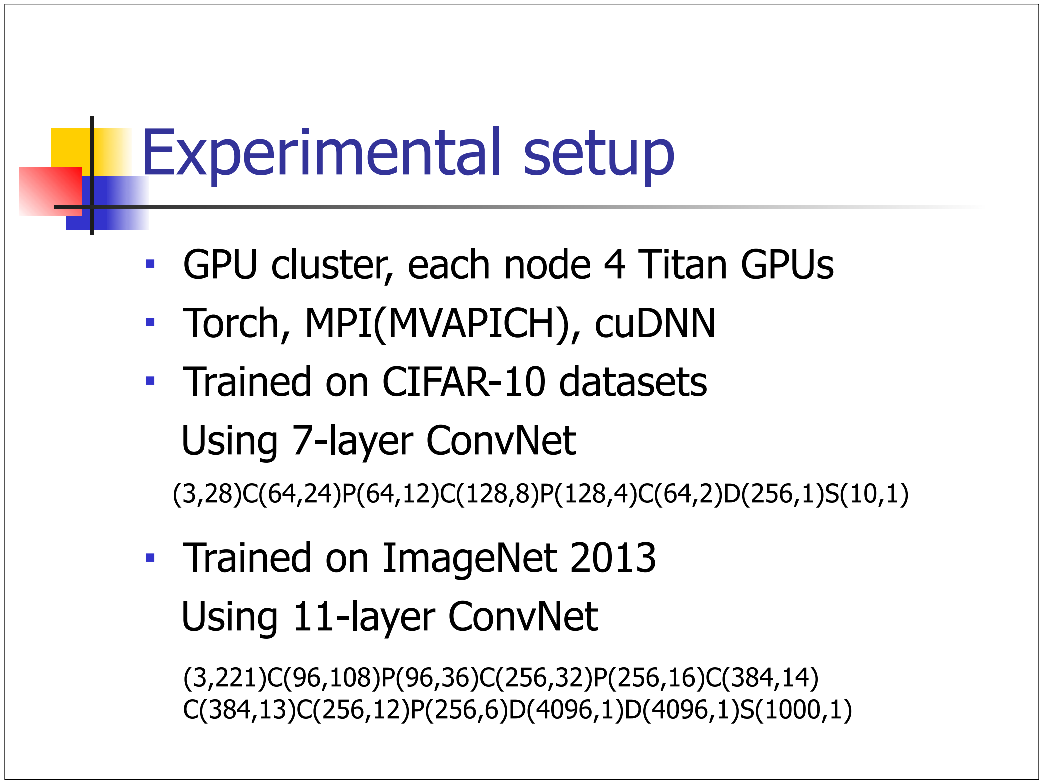
$$F^1_1 = \begin{pmatrix} 1 & -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, F^1_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{\eta\rho}{1+\eta\rho} & \frac{1-\eta}{1+\eta\rho} & 0 & 0 & \frac{\eta\rho}{1+\eta\rho} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, F^1_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{p} & \frac{1}{p} & -\frac{1}{p} & \frac{1}{p} & 0 \end{pmatrix}$$

$$F^1_3 \circ F^1_2 \circ F^1_1$$



The largest absolute value of eigenvalue of F should be inside unit circle on complex plane. ⟺ stable

This is another main contribution of the paper.
They show that they found a way to prove if an parameter updating scheme is stable.
Here’s an example parameter updating schem called ADMM.
Hadamard product element-wise.
This could help us tune hyper-parameters in an analytical way.



Experimental setup

- GPU cluster, each node 4 Titan GPUs
- Torch, MPI(MVAPICH), cuDNN
- Trained on CIFAR-10 datasets

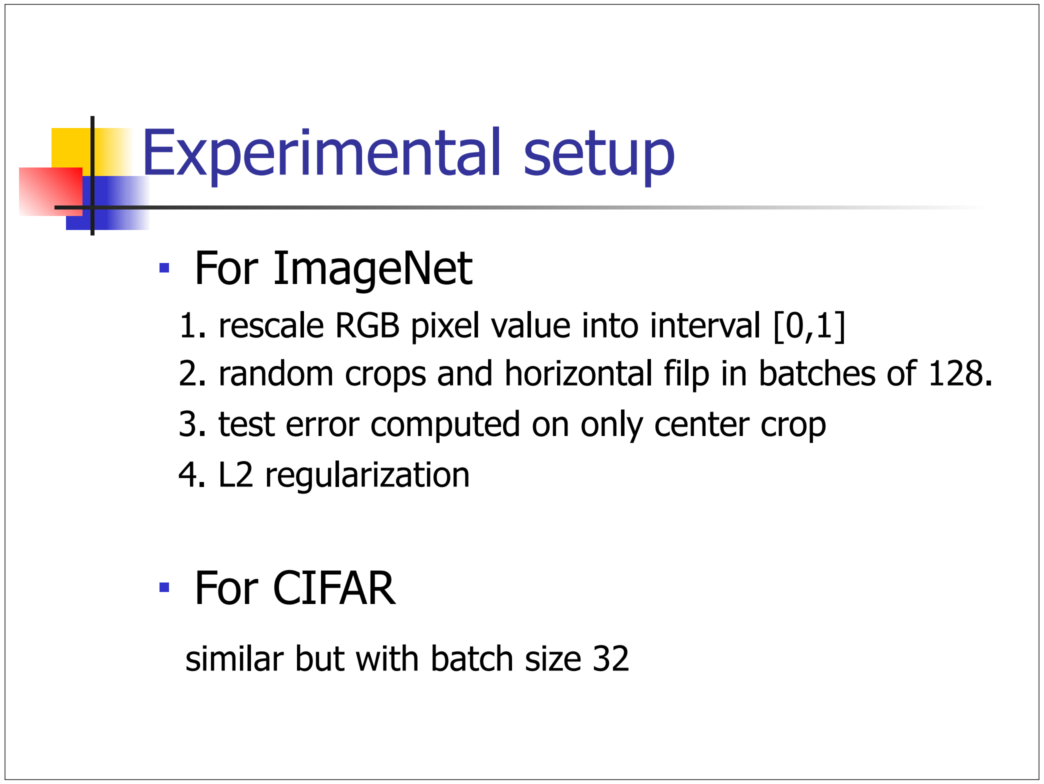
  Using 7-layer ConvNet

  (3,28)C(64,24)P(64,12)C(128,8)P(128,4)C(64,2)D(256,1)S(10,1)
- Trained on ImageNet 2013

  Using 11-layer ConvNet

  (3,221)C(96,108)P(96,36)C(256,32)P(256,16)C(384,14)
  C(384,13)C(256,12)P(256,6)D(4096,1)D(4096,1)S(1000,1)

Now lets talk about the experimental setup of their model,
GPU cluster
Should be cudnn2 or 3 as they start in Dec 2014 and updated in Aug 2015



Experimental setup

- For ImageNet

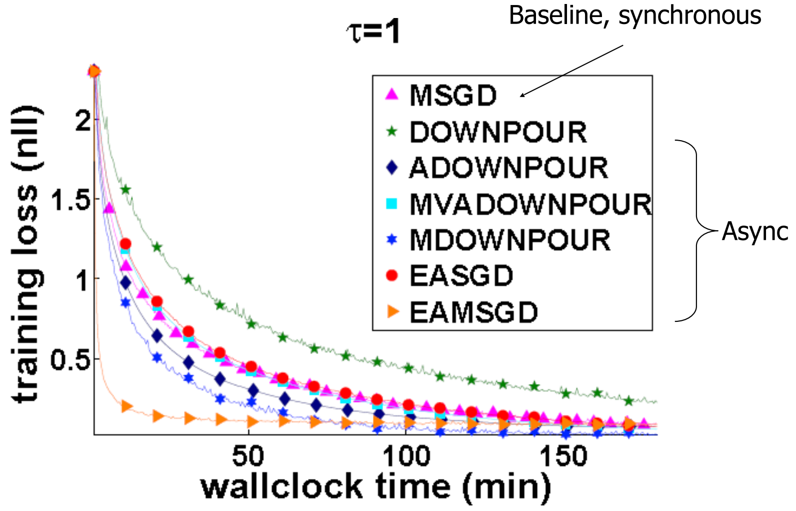
  1. rescale RGB pixel value into interval [0,1]
  2. random crops and horizontal filp in batches of 128.
  3. test error computed on only center crop
  4. L2 regularization

- For CIFAR

  similar but with batch size 32

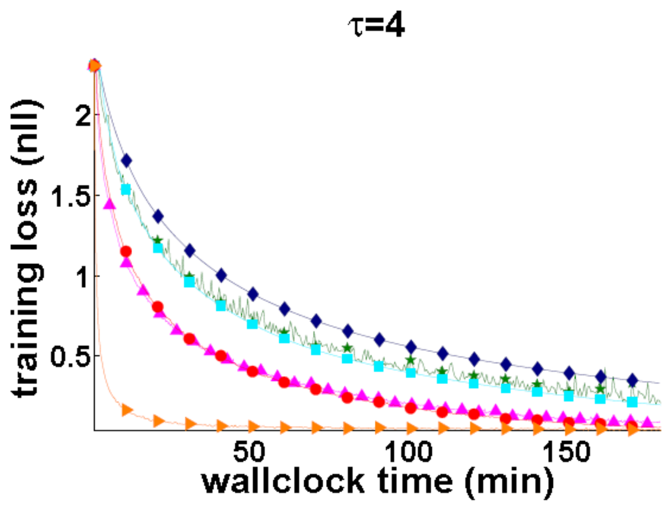
I guess they do these would protentially cost more space usage on GPU at the input level.

Results: sensitivity of tau

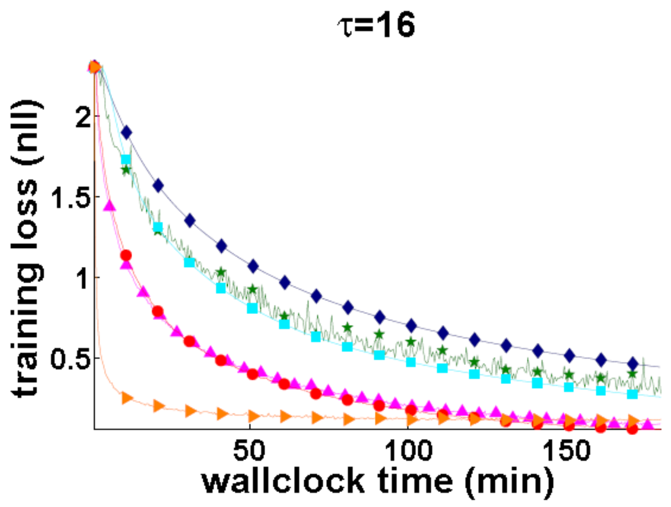


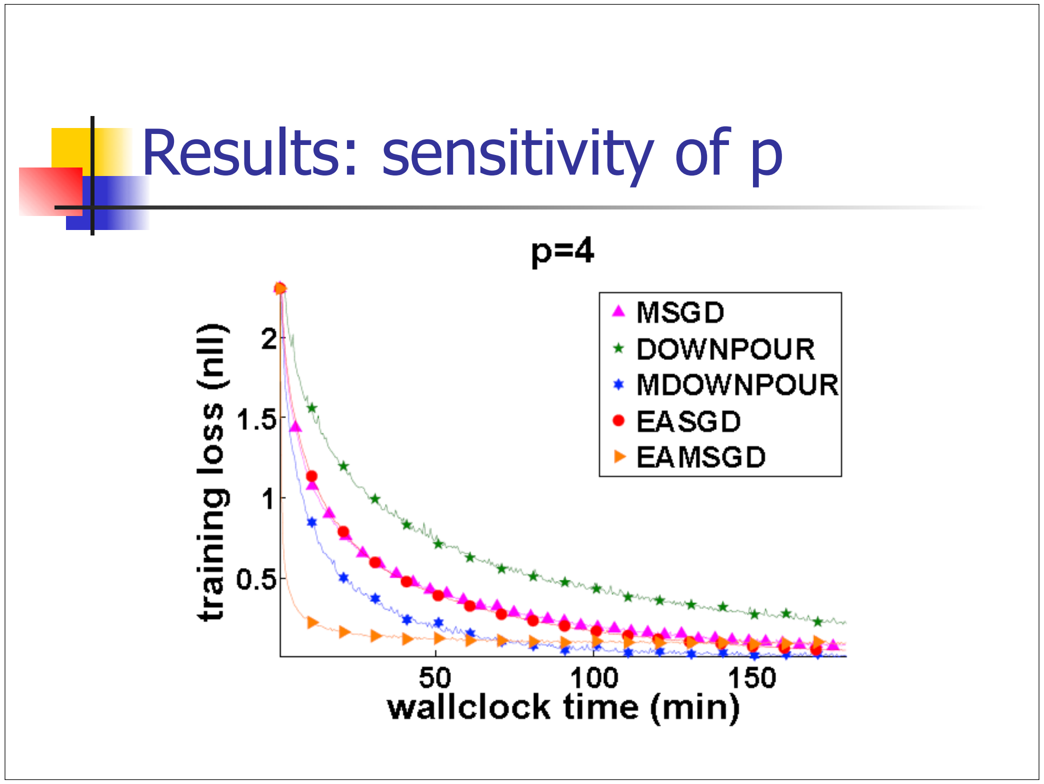
The baseline is the MSGD which is sequential algorithm (there is no master, purely synchronous) so it doesn't change with tau.
It can be seen that the EASGD and EAMSGD stick with the baseline closely no matter how large the communication period is. While other algorithms deviate from baseline.
And the EAMSGD always provide faster convergence compared to other asynchronous algorithms.
This is trained on CIFAR.

Results: sensitivity of tau

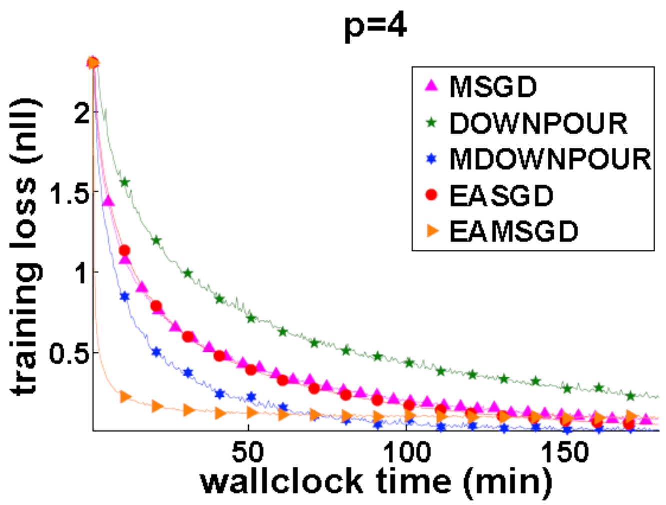


Results: sensitivity of tau



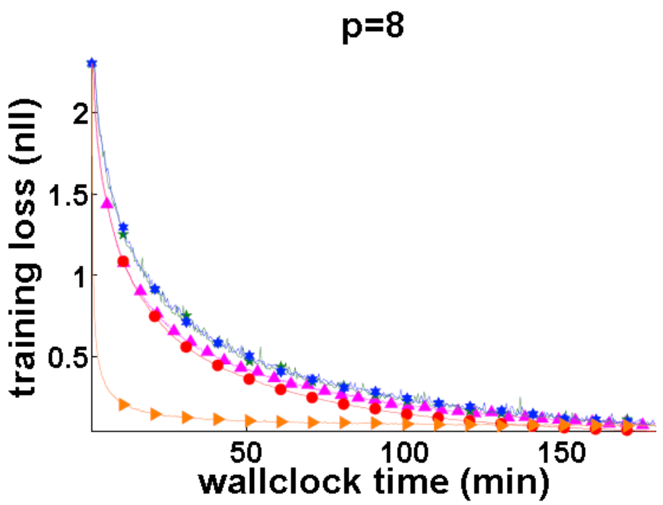


Results: sensitivity of p

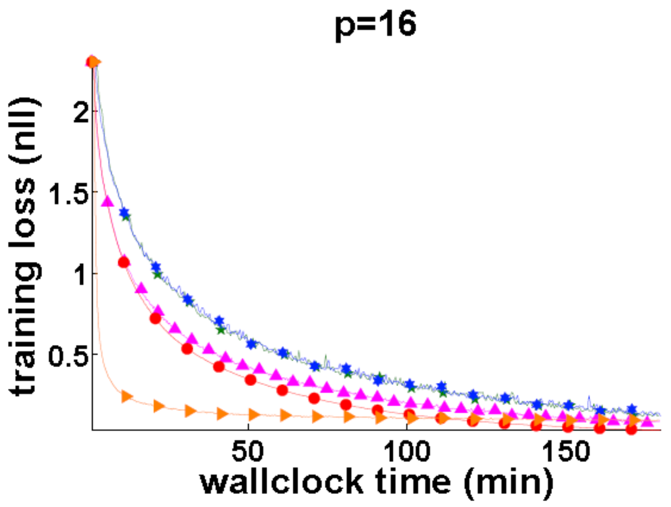


It can be seen that both down pour and mdownpour converge worse when number of workers increases.

Results: sensitivity of p

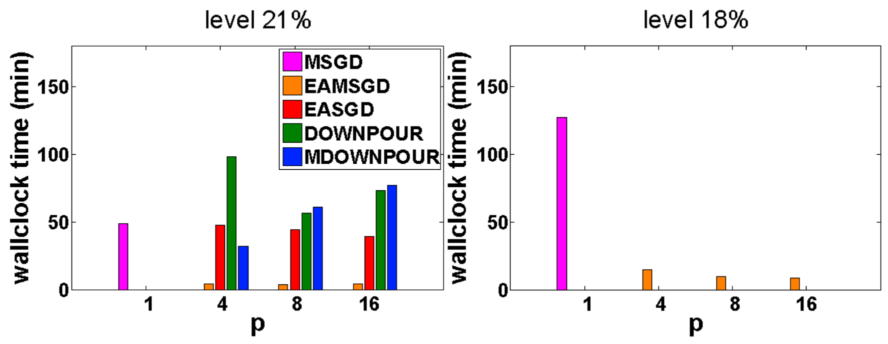


Results: sensitivity of p



Results: time speed up

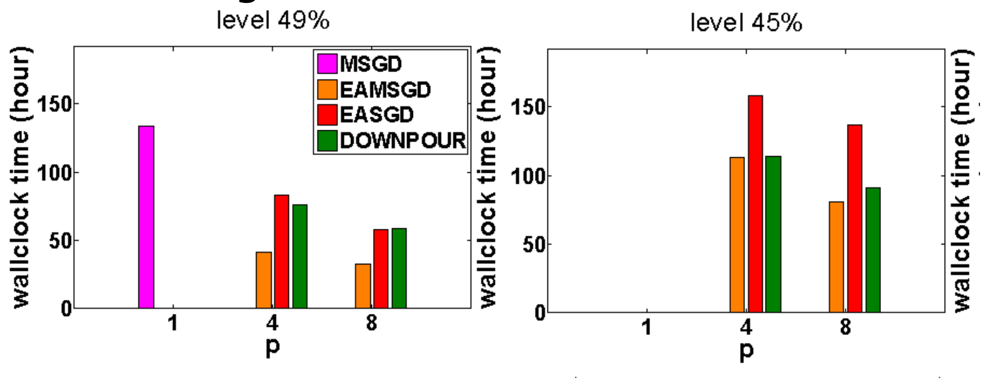
- CIFAR



Here we can see the EASGD red can maintain almost the same performance level and a little bit decreasing but not linearly.
Also we can see that the performance of google Downpour algorithm seems bottlenecked when using two many process (16 vs. 8), this is possibly from communication.
Aother explanation from the paper is that lr chosen is based on the smallest achievable test error. So the more processes, the less learning rate.

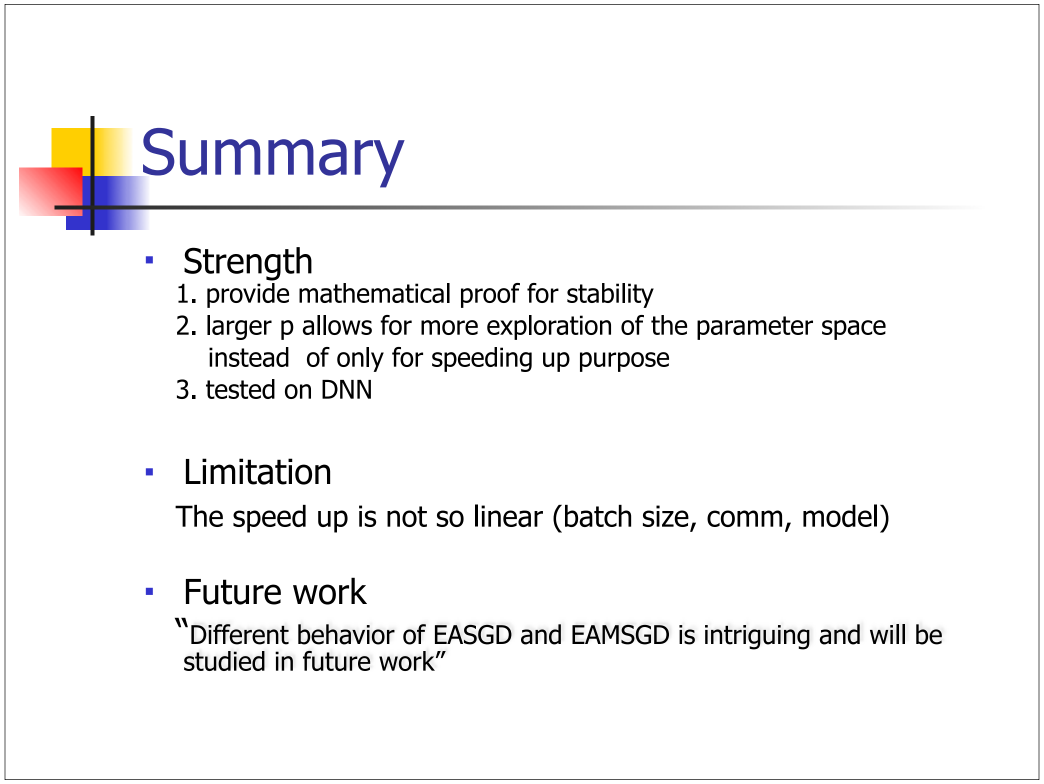
Results: time speed up

- ImageNet



Our level of BSP reaching val error==49% is 25 epochs * 0.52 h/epoch = 13h
Our level of BSP reaching val error==45% is 45 epochs * 0.52 h/epoch = 23h

But this is not comparable to their result, maybe because the speed difference of communication and GPUs, maybe they augment data to achieve lower convergence.



Summary

- Strength
  1. provide mathematical proof for stability
  2. larger p allows for more exploration of the parameter space instead of only for speeding up purpose
  3. tested on DNN

- Limitation

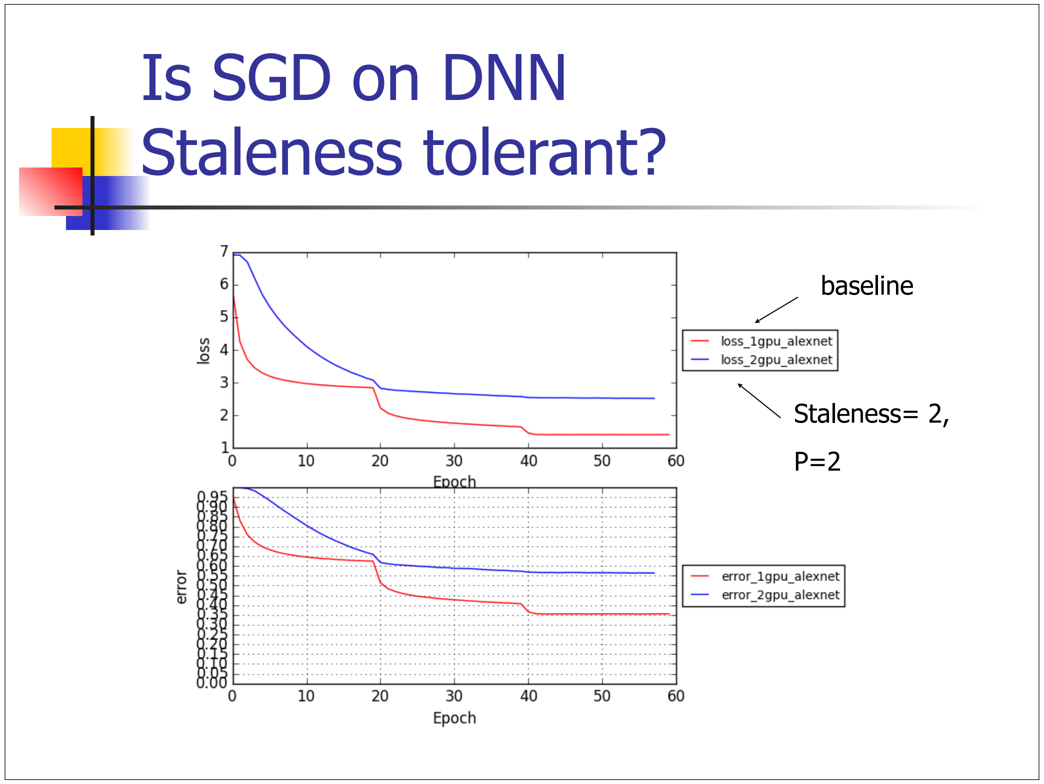
  The speed up is not so linear (batch size, comm, model)

- Future work

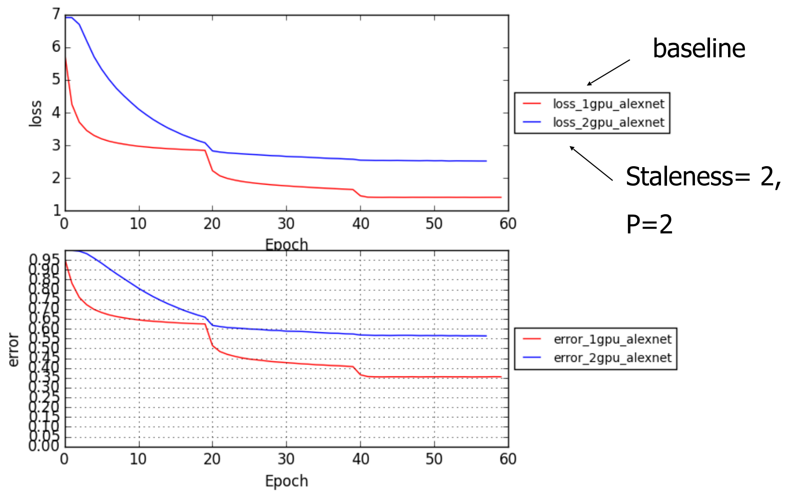
  "Different behavior of EASGD and EAMSGD is intriguing and will be studied in future work"

They didn't provide source code for readers.
Speed up is not so linear: could be small batch size(32), could be high communication overhead, could be the model it self.
For our future work, we could potentially try out their method to prove the stablility of BSP in our structure and get some ranges of hyperparameters.
We could built EASGD based on Theano and compare the results with BSP.

Questions

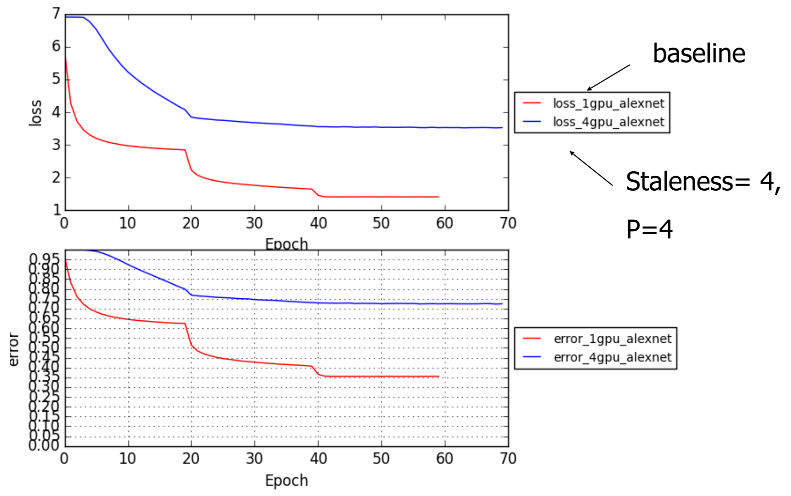


Is SGD on DNN Staleness tolerant?



Also the async SGD and SSP algorithms are based on its staleness tolerance.
That is to say, if the SGD scheme and certain types of models can not tolerant staleness, the model won't converge well.
This can be equivalent to delay the weight updates for a few iteration on BSP.

Is SGD on DNN Staleness tolerant?



This experiment is not finished. I should do more test. But this figure at least these shows that the DNN can tolerate some staleness. If let the first step be longer. It seems it would converge better. And convergence is related to number of process or Staleness.