Predicting Statistics of Asynchronous SGD Parameters for a Large-Scale Distributed Deep Learning System on GPU Supercomputers

Yosuke Oyama[1,a)] Akihiro Nomura[1] Ikuro Sato[2] Hiroki Nishimura[3] Yukimasa Tamatsu[3] Satoshi Matsuoka[1]

1 Tokyo Institute of Technology

2 DENSO IT LABORATORY, INC.

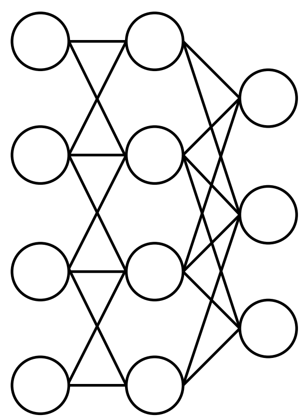
3 DENSO CORPORTATION

a) oyama.y.aa@m.titech.ac.jp

Background
Deep Learning (DL)

- A machine learning technique using "Deep" Neural Network
  - DL is achieving state-of-the-art in large machine learning area
  - Training DNN with huge dataset requires large scale computation
    - eg. 15-layer CNN training takes 8.2 days on 16 nodes (48 GPUs) of TSUBAME2.5
    - Researchers have to train DNN for several times to optimize DNN structure and hyper-parameters by hand

Imput (Image)
(Reference: http://image-net.org/)



Neural Network

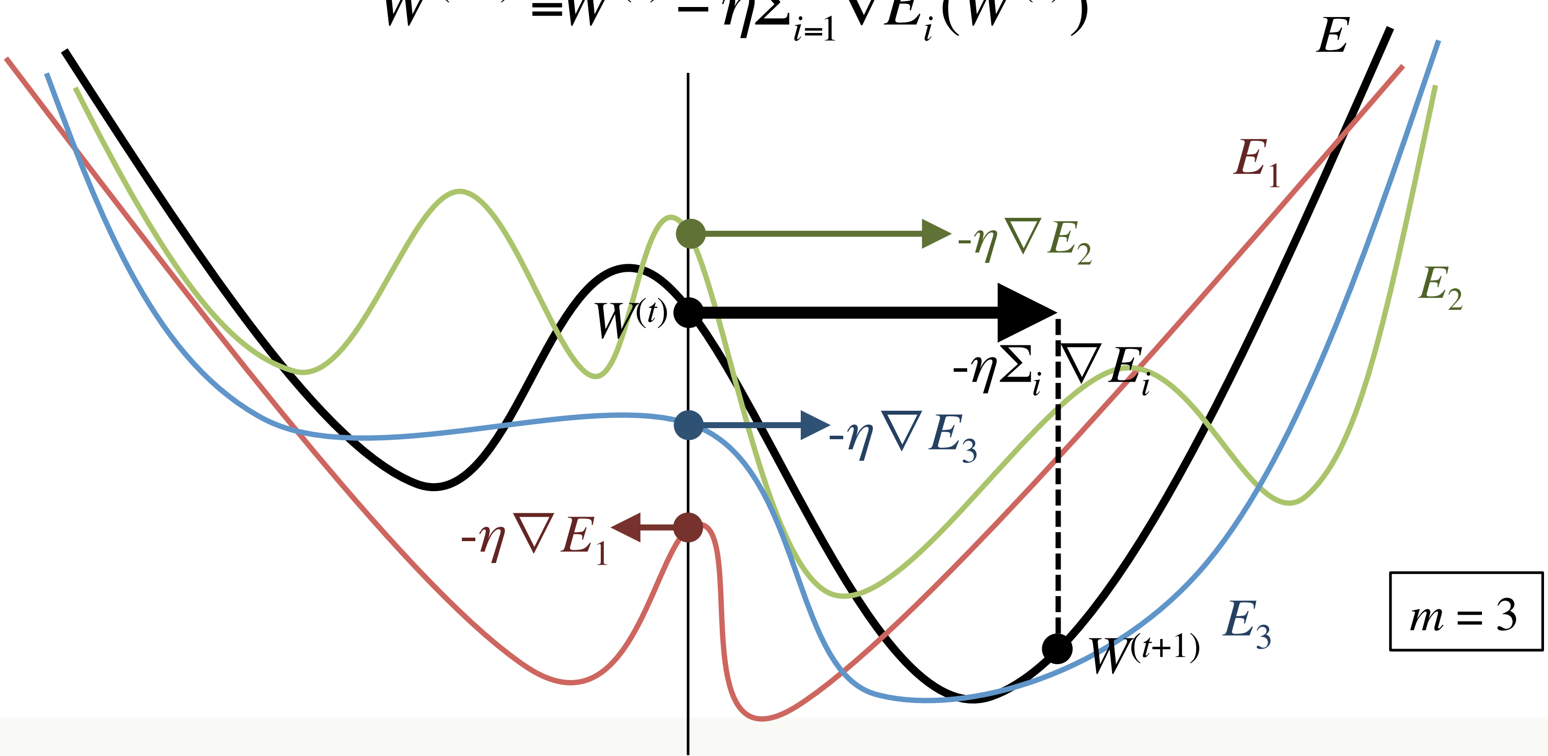
Barn swallow = **0.95**
Police dog = 0.03
Water beetle = 0.01
⋮

Output (Classification)

Background
Stochastic Gradient Descent

- An optimization method to update NN weights $W^{(t)}$ with summation of gradient ∇E_i of m samples (i.e. mini-batch)
  - Suitable for DL, in which computing global gradient ∇E requires hundreds PFLOP

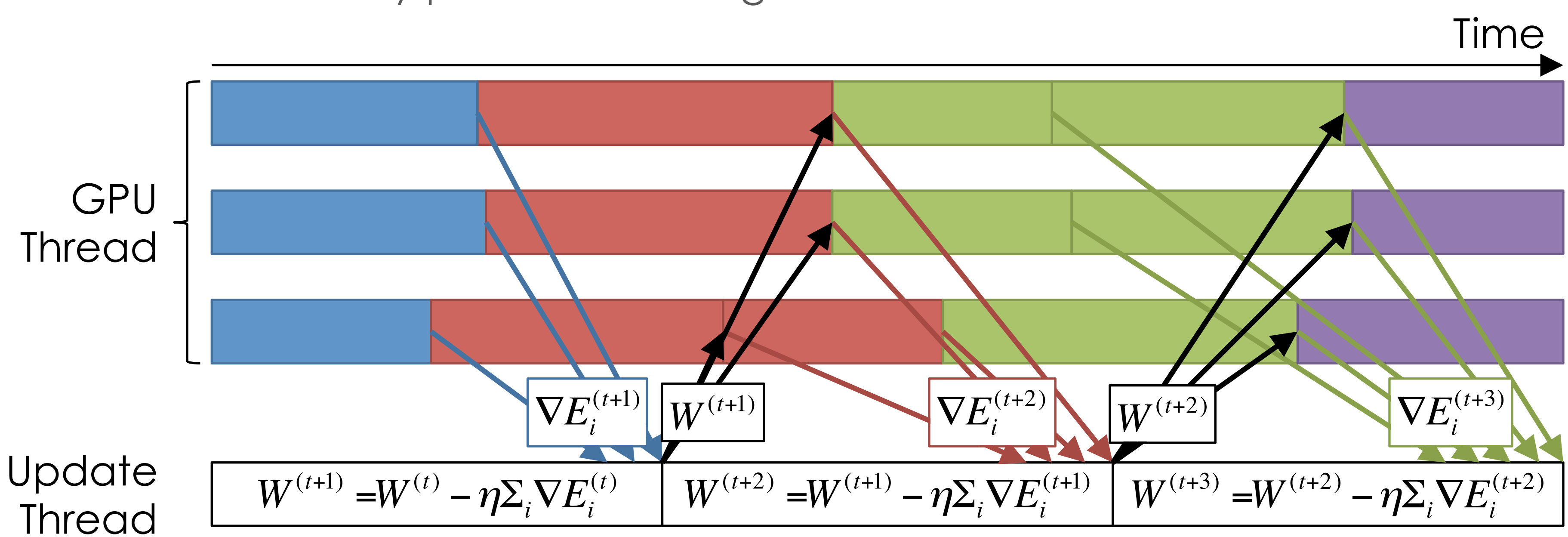
$$W^{(t+1)} = W^{(t)} - \eta \Sigma_{i=1}^{m} \nabla E_i(W^{(t)})$$



Background

Asynchronous Stochastic Gradient Descent (ASGD)

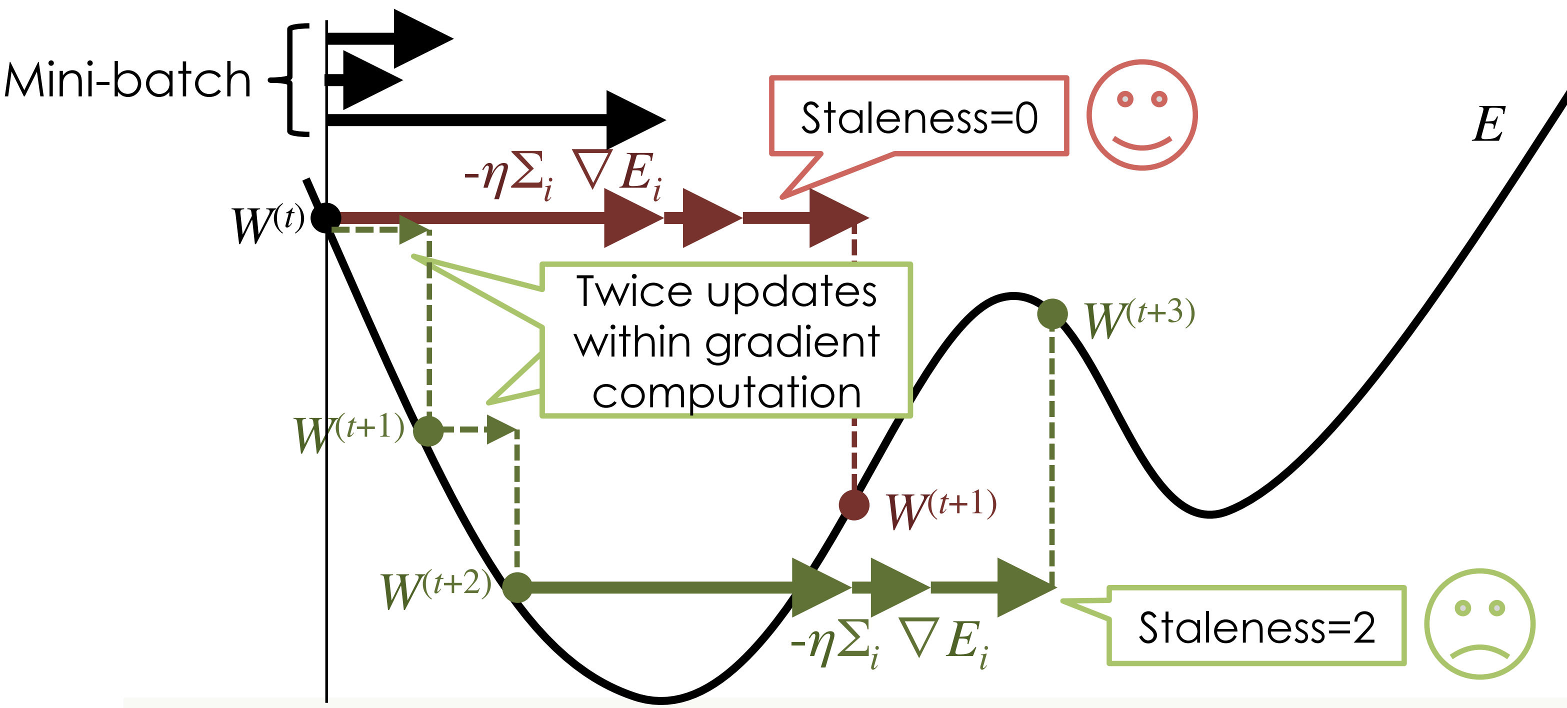
- **GPU threads** independently compute gradient of distinct samples, while **update threads** update DNN weights asynchronously
  - ASGD may speed-up the training
  - ASGD may produce worse generalization error



Timeline of ASGD Training

Background
Mini-batch Size and Staleness

- **Staleness**: # of updates done within one gradient computation
- Existing researches showed that the error is increased by larger mini-batch size and staleness
  - **There was no way of knowing these statistics in advance**

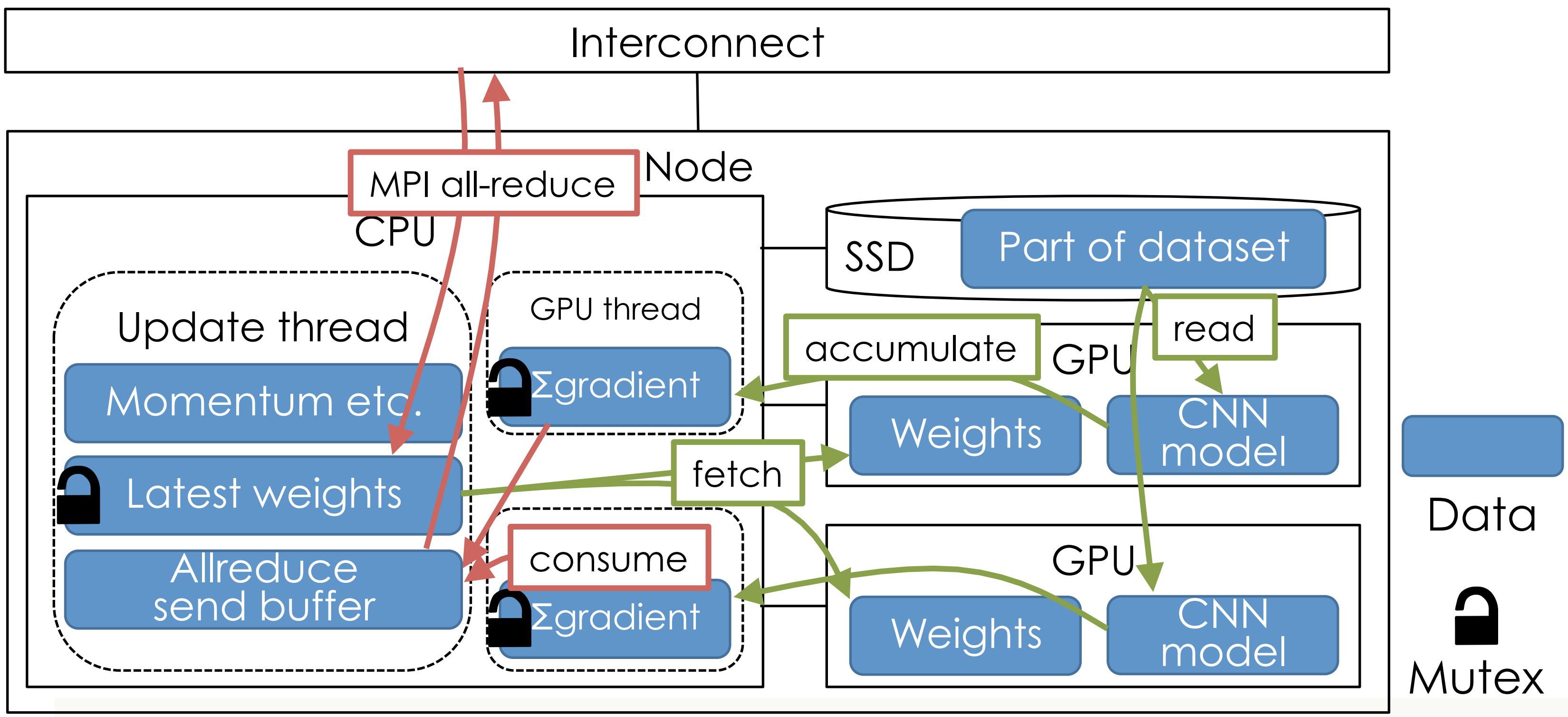


Approach and Contribution

- **Approach**: Proposing a performance model for an ASGD deep learning system, which considers probability distribution of mini-batch size and staleness
  - Takes CNN structure and machine specifications as input
  - Predicts time to sweep entire dataset (epoch time) and the distribution of the statistics
- **Contribution**
  - Our model predicts epoch time, average mini-batch size and staleness with 5%, 9%, 19% error in average respectively on several supercomputers
  - Our model steadily choose the fastest machine configuration that nearly meets a target mini-batch size
  - Our model predicts how DL scales with upcoming hardware specification
    - FP16, EDR InfiniBand

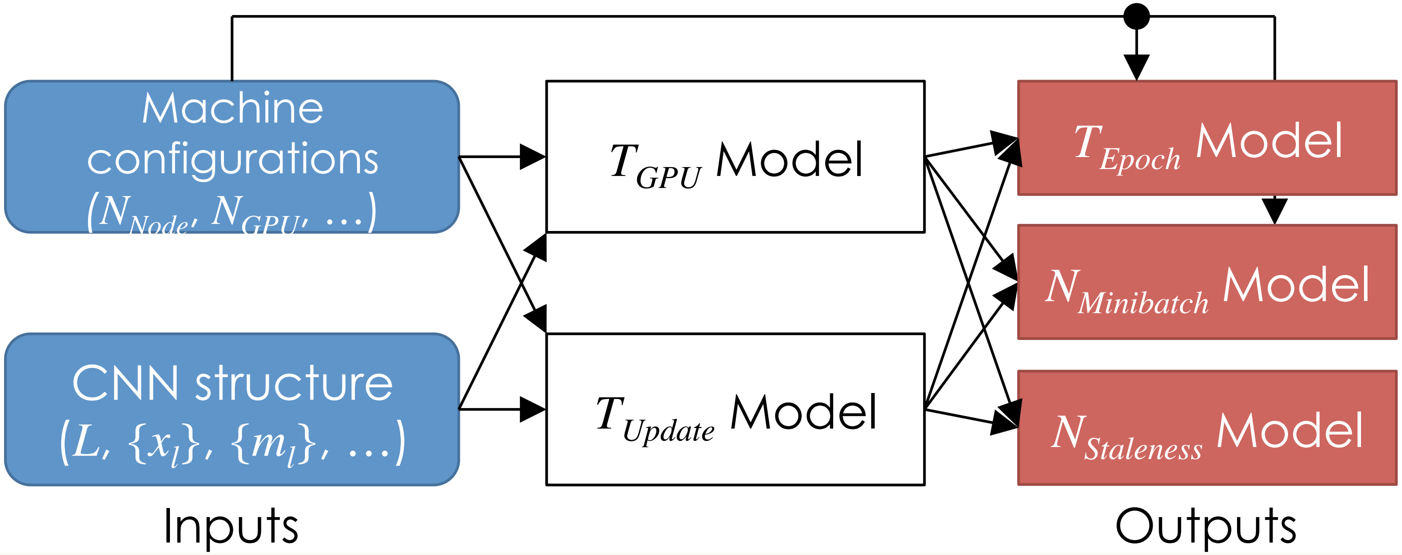
SPRINT Overview

- SPRINT is a data-parallel ASGD application to train CNN with GPUs
  - GPU threads compute gradient of randomly-picked samples and accumulate it to the host memory
  - Update threads execute MPI all-reduce to update the weights



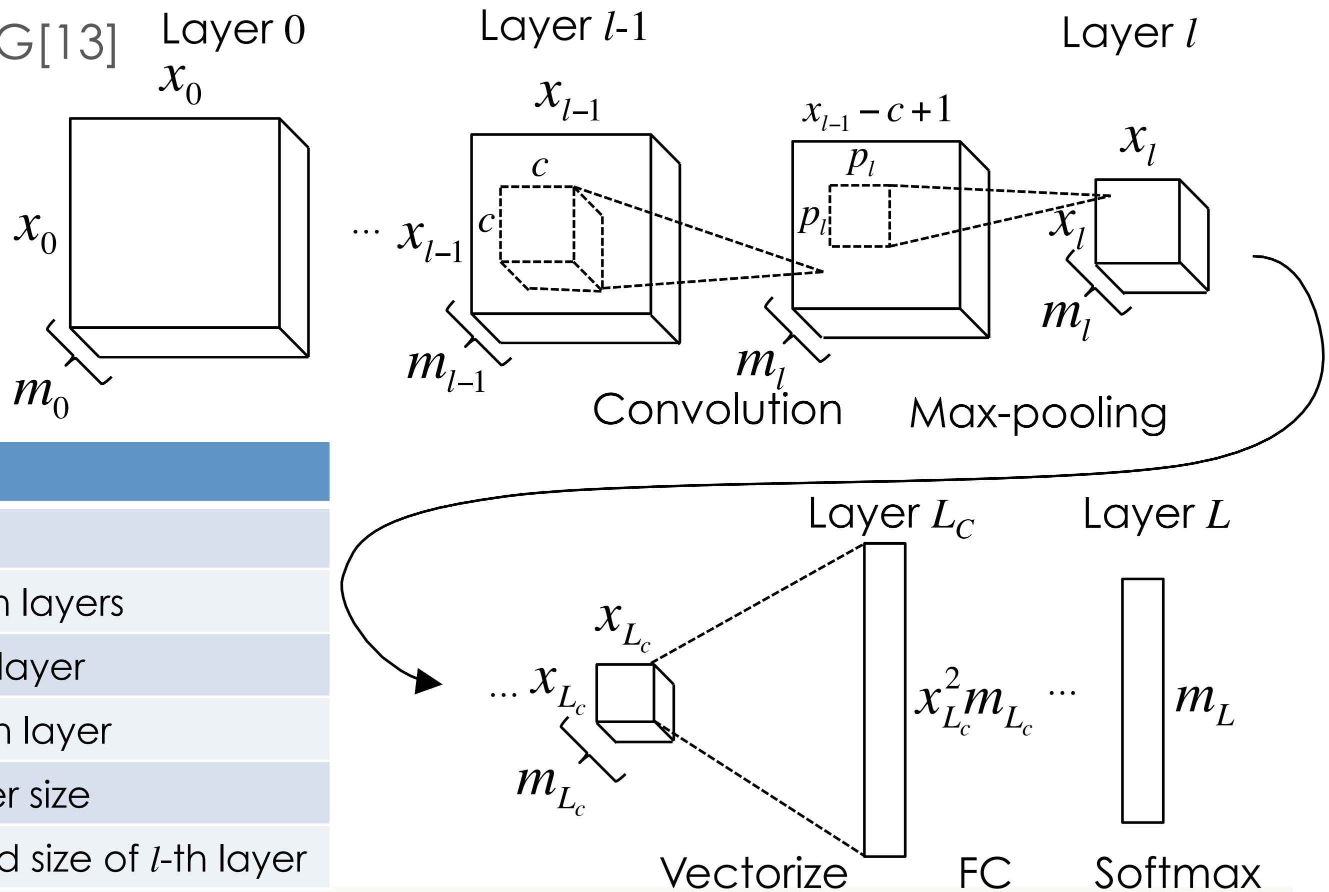
Proposed Performance Model
Overview

1. Takes # of nodes (N_{Node}), # of GPUs (N_{GPU}), CNN structure as input parameters
2. Predicts execution time of one iteration of GPU threads and update threads (T_{GPU}, T_{Update})
3. Predicts
   - epoch time (T_{Epoch}) as a constant
   - Mini-batch size ($N_{Minibatch}$) and staleness ($N_{Staleness}$) as stochastic variables



Proposed Performance Model
CNN Structure

- The model supports CNNs with convolution layers, optional max-pooling and fully-connected layers
  - Example: VGG[13]

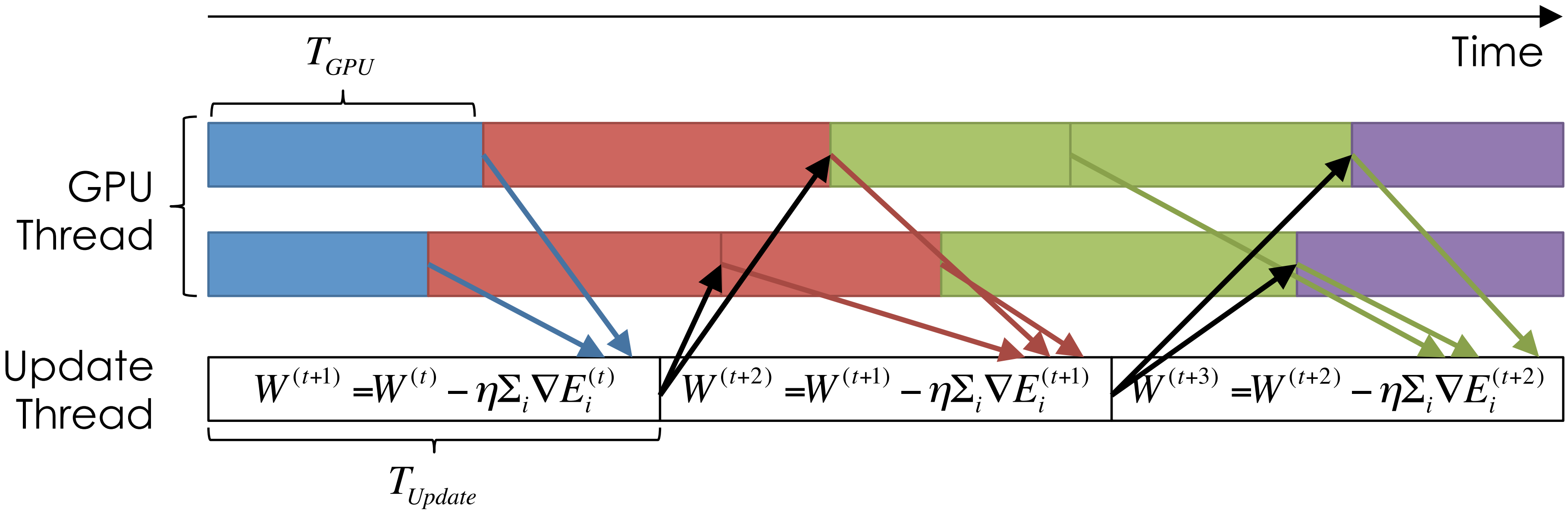


Parameter	Meaning
L	# of all layers
L_C	# of convolution layers
x_l	Map size of l-th layer
m_l	# of maps of l-th layer
c	Convolution filter size
p_l	Max-pooling grid size of l-th layer

Proposed Performance Model
Execution Time of Thread Iteration

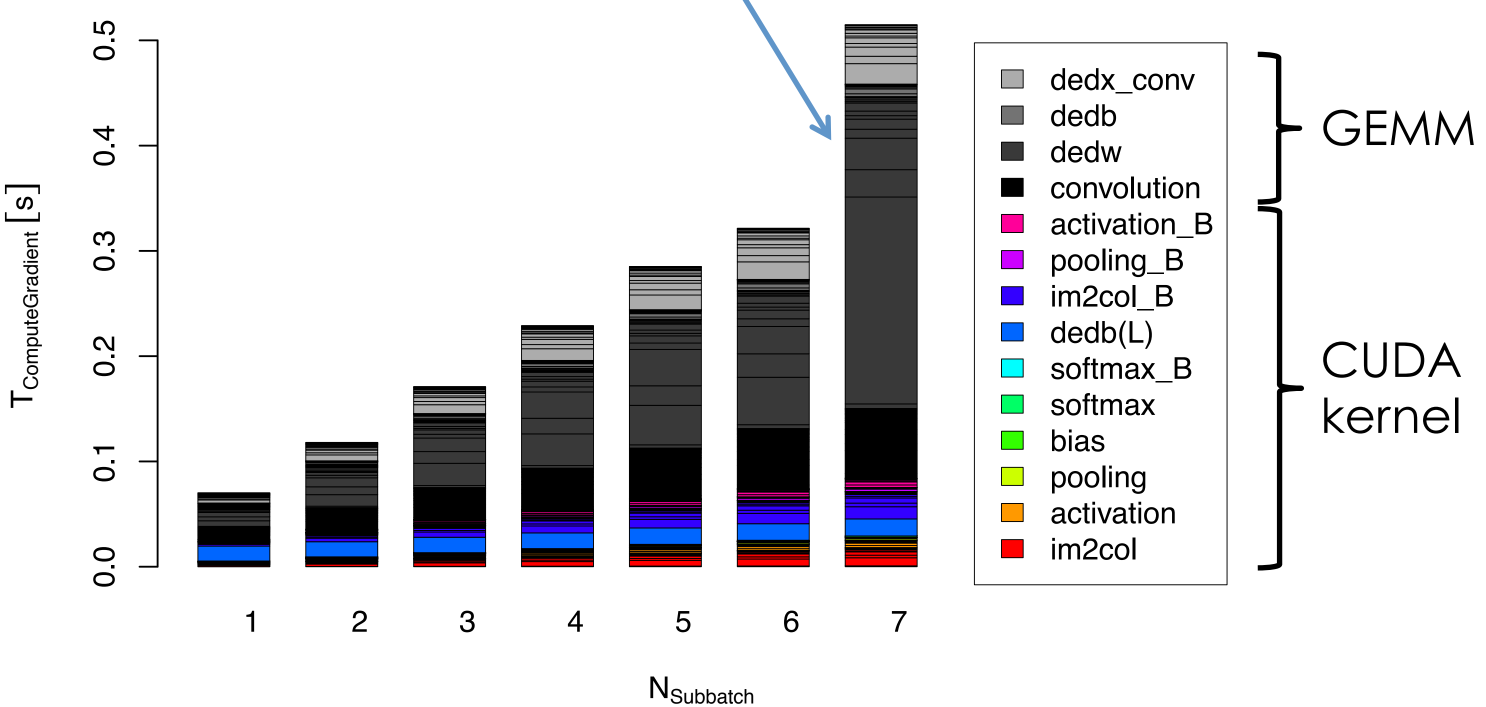
- Execution time of thread iteration is divided into several tiny sub-models, each representing time complexity of its part
  - Coefficients are fitted with the least squares method

$$T_{GPU} = T_{LoadImage} + T_{ComputeGradient} + T_{UpdateGradient} + \dots$$
$$T_{Update} = T_{SumGradient} + T_{Allreduce} + T_{UpdateWeights} + \dots$$



Proposed Performance Model
Execution Time of Gradient Computation

- One gradient computation iteration is consisted of various CUDA kernels and SGEMM
  - 15-layer CNN calls more than 100 kernels/GEMMs per iteration



Breakdown of Measured Gradient Computation Time of 15-layer CNN on NVIDIA Tesla K80

Proposed Performance Model
Execution Time of Gradient Computation

- Computation time is modeled with summation of consisting kernels

$$T_{ComputeGradient} = \sum_{l=1}^{L_c} \{T_{im2col}(l) + T_{convolution}(l) + T_{activation}(l)\} + \cdots$$

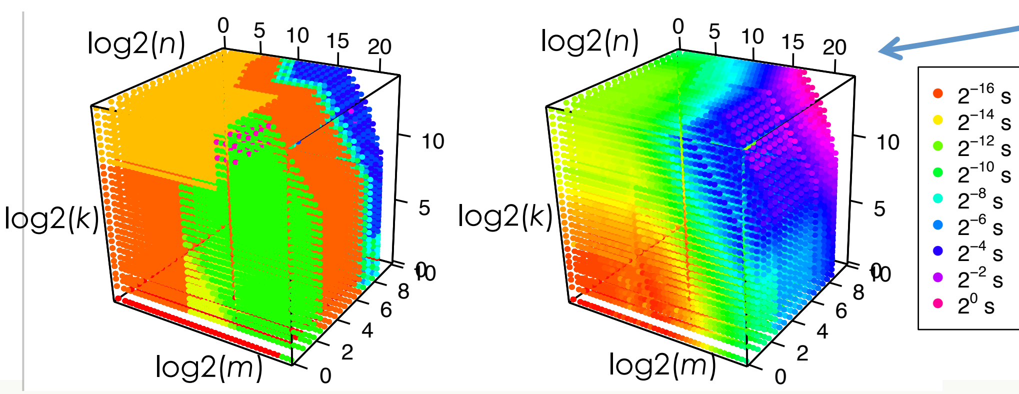
- **Model for CUDA Kernel:** Linear function of its computation complexity

$$T_{im2col}(l) = \alpha x_l'^2 c^2 m_{l-1} N_{Subbatch} + \beta$$

Coefficients are fitted with the least square method

- **Model for GEMM:** Interpolation of measured computation time

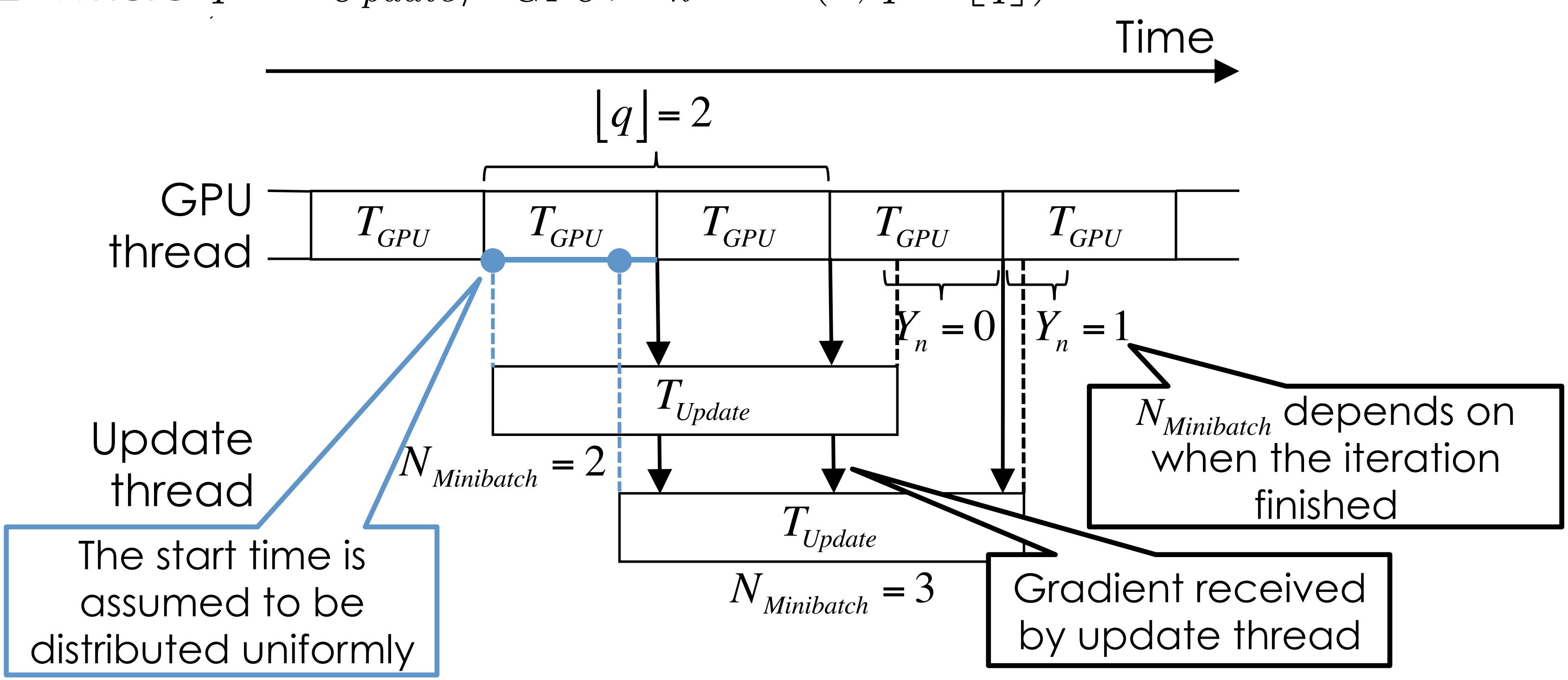
$$T_{convolution}(l) = \sum_{m,n,k \in \{0,1\}} \alpha_{m,n,k} (x_l'^2 N_{Subbatch})^m m_l{}^n (c^2 m_{l-1})^k$$



Combination of Internal Kernels (Left) and Computation time (Right) of *cublasSgemm* on NVIDIA Tesla K80

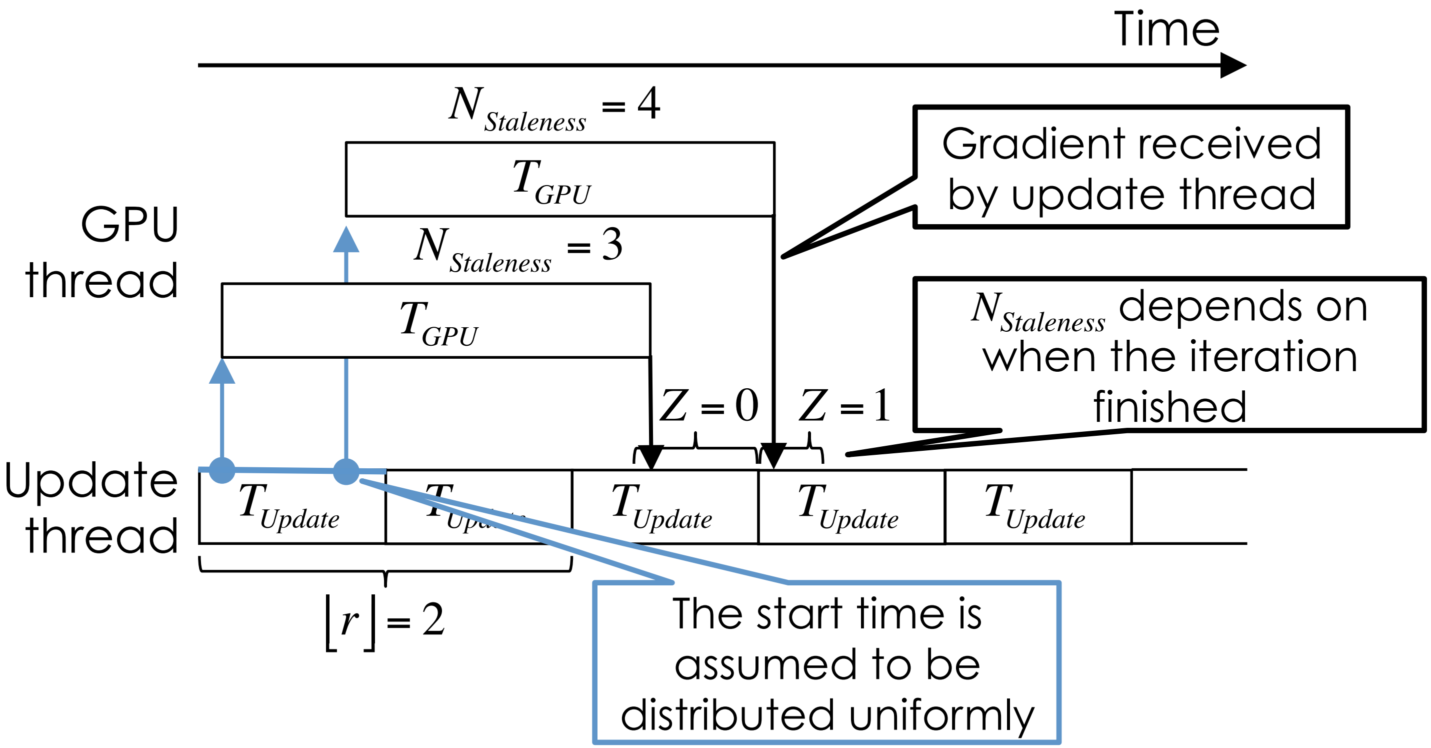
Proposed Performance Model
Predicting Distribution of Mini-batch Size

- $$N_{Minibatch} = N_{Subbatch} \sum_{n=1}^{N_{Node} \times N_{GPU}} (Y_n + \lfloor q \rfloor)$$
- Where $q = T_{Update}/T_{GPU}$, $Y_n \sim B(1, q - \lfloor q \rfloor)$



Proposed Performance Model
Predicting Distribution of Staleness

- $N_{Staleness} = Z + \lfloor r \rfloor + 1$
  - Where $r = T_{GPU}/T_{Update}$, $Z \sim B(1, r - \lfloor r \rfloor)$



Proposed Performance Model

Predicting Average Statistics

- Three outputs are computed from thread iteration time and machine configurations

# of Samples

$$T_{Epoch} = \frac{\boxed{N_{File}} \times \boxed{T_{GPU}}}{\boxed{N_{Node} \times N_{GPU}} \times \boxed{N_{Subbatch}}}$$

# of GPUs

Time to process one sample on a GPU

$$\overline{N_{Minibatch}} = \frac{\boxed{N_{Node} \times N_{GPU}} \times \boxed{N_{Subbatch}} \times \boxed{\overline{T_{Update}}}}{\boxed{T_{GPU}}}$$

# of GPUs

Time to update weights

Frequency to process one sample on a GPU

$$\overline{N_{Staleness}} = \boxed{T_{GPU} / \overline{T_{Update}}} + \boxed{1}$$

Staleness due to Non-blocking update

Staleness due to Slow GPU iteration

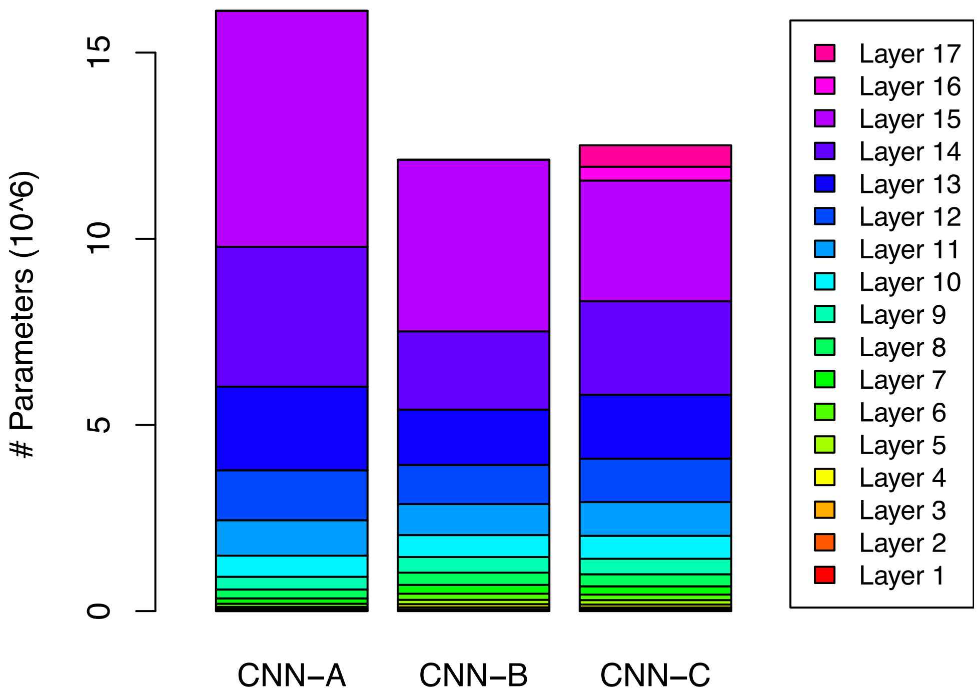
Evaluation

- The proposed performance model is evaluated on TSUBAME 2.5 and TSUBAME-KFC/DL
  - Up to 64 nodes of TSUBAME 2.5, or 16 nodes of TSUBAME-KFC/DL are used for evaluation

	TSUBAME 2.5	TSUBAME-KFC/DL
# nodes	1408	42
CPU	Intel Xeon X5670 x 2	Intel Xeon E5-2620v2 x 2
GPU	NVIDIA Tesla K20X x 3	NVIDIA Tesla K80 x 4
Network	4X QDR InfiniBand x 2	4X FDR InfiniBand
Compiler	ICC 14.02	ICC 14.0.0
CUDA	CUDA 7.0	
MPI	MVAPICH2 2.0rc1	

Evaluation

- Three 15-17 layers CNNs are used for evaluation
  - Coefficients of the model are fitted with CNN-A and subsets of training configurations (N_{Node}, $N_{Subbatch}$)
  - Prediction error is measured with CNN-A, B, and C
  - ILSVRC2012 dataset is used for evaluation



Breakdown of # parameters of three CNNs

	CNN-A	CNN-B	CNN-C
Input layer size	396	396	346
# con[illegible]	[illegible]	15	17
# max-p[illegible]ing	5	5	5
# parameters	16.1 M	12.1 M	12.5 M
GEMM GFLOP per sample	41.0	100.0	184.5

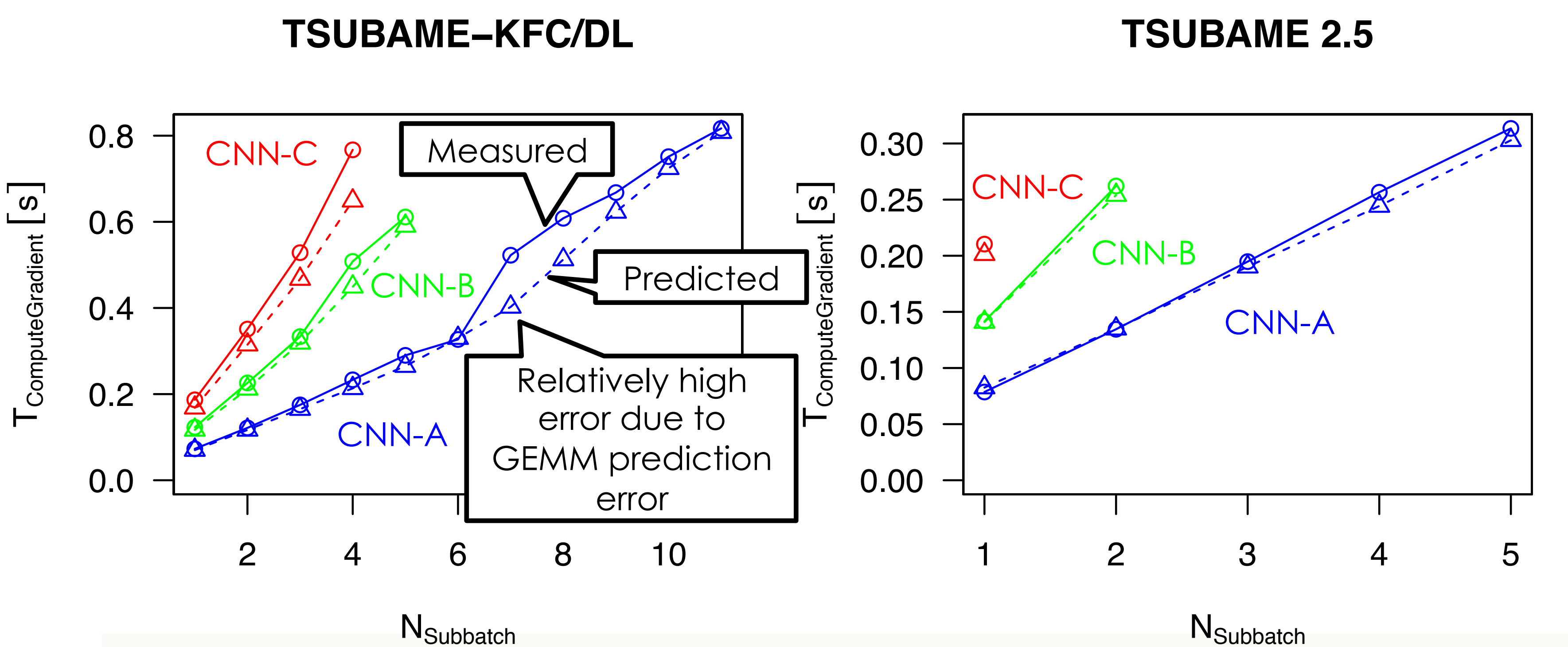
$O(\Sigma_l m_l^2 c_l^2)$

$O(\Sigma_l m_l^2 x_l^2)$

Evaluation
Execution Time of Gradient Computation

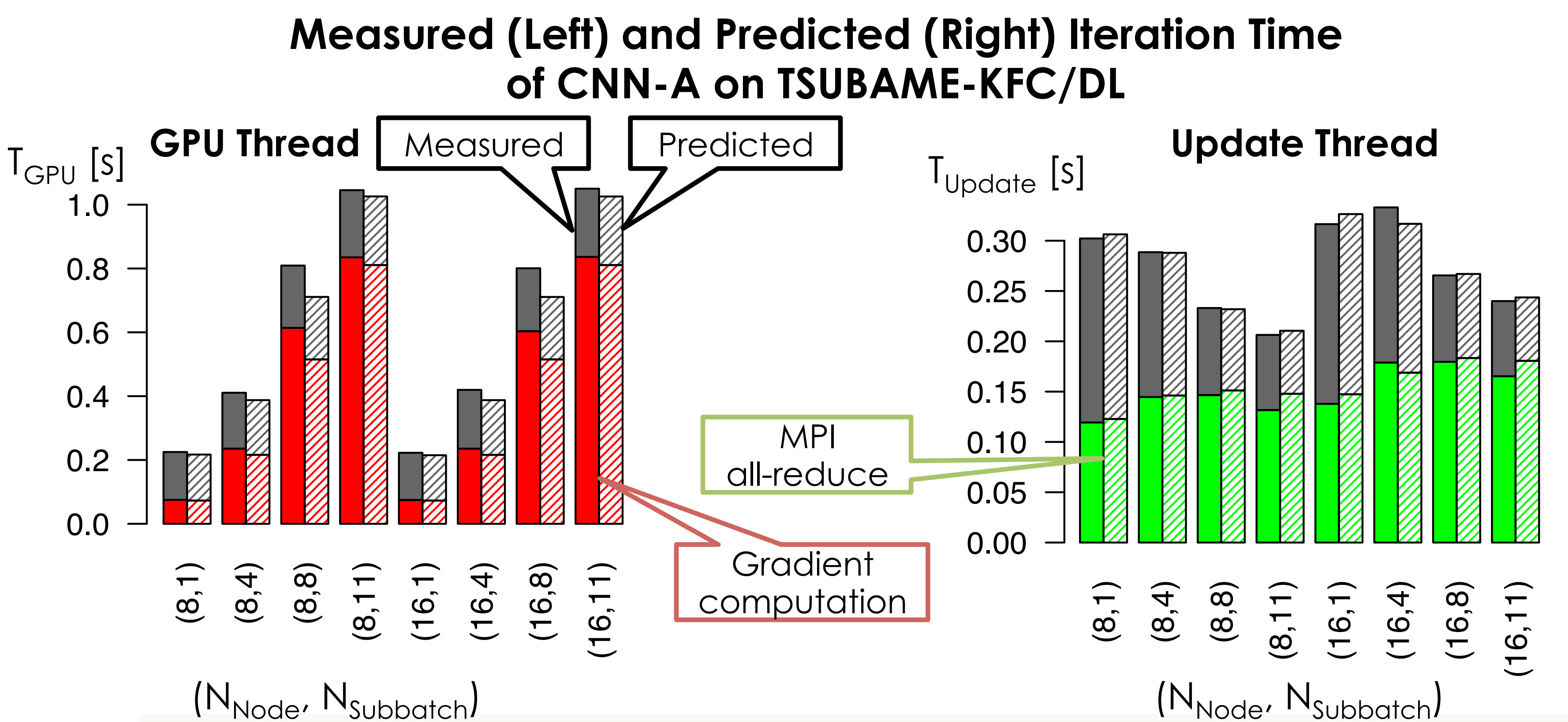
- In all CNNs and $N_{Subbatch}$, the prediction error was lower than 12% on average

Measured (Solid) and Predicted (Dashed) Gradient Computation Time of Three CNNs on Two GPUs



Evaluation
Iteration Time of GPU/Update Thread

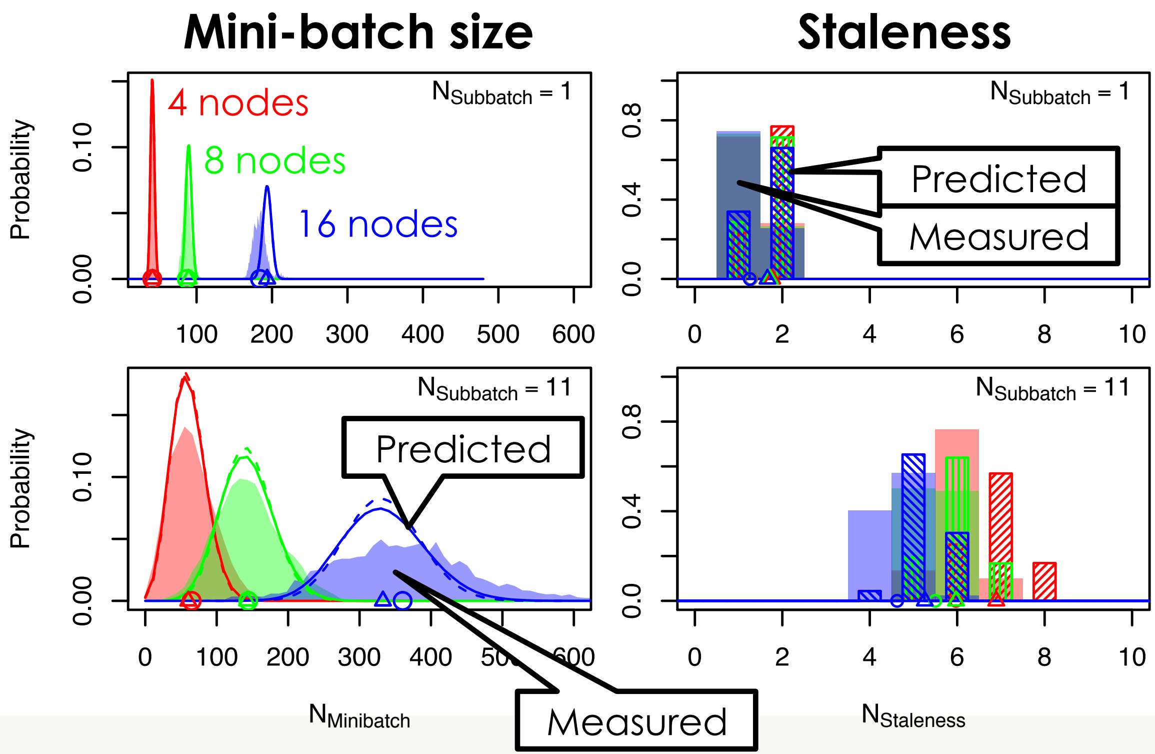
- The prediction error of GPU thread iteration time was less than 7%, 11% on average respectively



Evaluation
Distribution of Mini-batch Size and Staleness

- Our model successfully predicts the distribution of mini-batch size and staleness
  - The average prediction error was 9% and 19% respectively

Distribution of mini-batch size and staleness of CNN-A on TSUBAME-KFC/DL



Evaluation

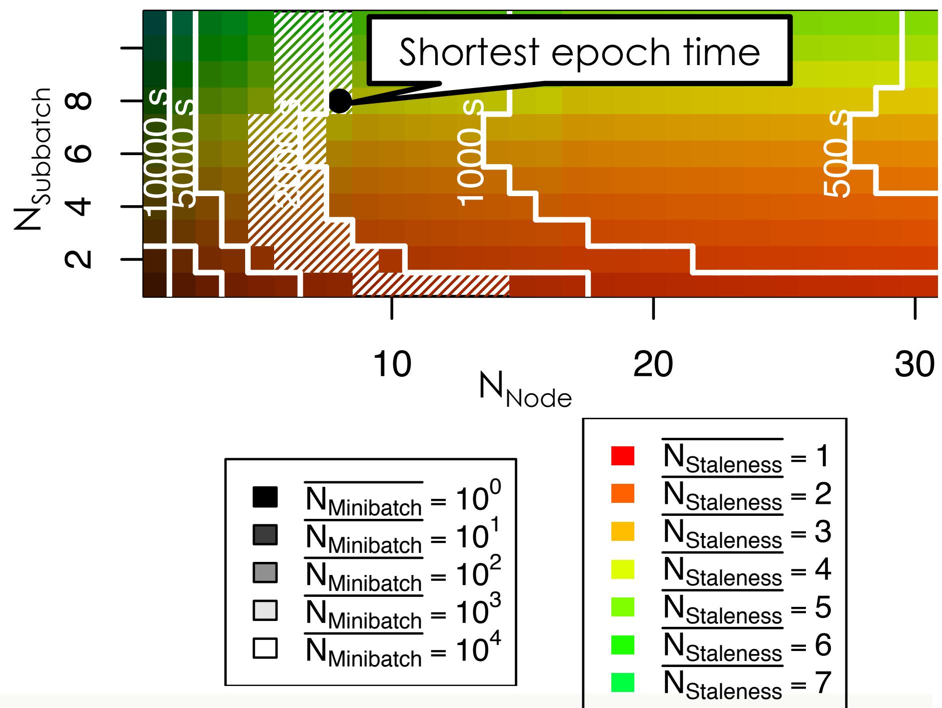
Parameter Search for Target Mini-batch Size

- Our model steadily chose the fastest configuration from subset for arbitrary target mini-batch size 138±25%

Predicted Ranking of Configuration for Shortest Epoch Time[s] of CNN-A on Two Supercomputers

	N_{Node}	$N_{Subbatch}$	Measured	Predicted
KFC	8	8	2025	1779
KFC	8	11	2316	2226
T2.5	16	5	2725	2614
T2.5	16	4	2910	2840
T2.5	32	1	3178	3227
T2.5	16	3	3276	3257

Predicted Epoch Time (Contour), Mini-batch Size (Brightness) and Staleness (Hue) of CNN-A on TSUBAME-KFC/DL



Evaluation
Performance Prediction for Future Hardware

- We predicted the optimal configurations within mini-batch size 138±25% for future hardware improvement
  - **FP16**: Computation time is halved and max $N_{Subbatch}$ is doubled
  - **EDR IB**: Communication time of all-reduce is divided by 12.5/7
    - 4xFDR InfiniBand (7 GB/s) → 4xEDR InfiniBand (12.5 GB/s)
- Interconnect performance is important as well as GPU performance to accelerate DL

The Optimal Predicted Configurations of CNN-A on TSUBAME-KFC/DL

	N_{Node}	$N_{Subbatch}$	Average mini-batch size	Epoch time[s]	Speedup
Baseline	8	8	165.1	1779	-
FP16	7	22	170.1	1462	**1.22**
EDR IB	12	11	166.6	1245	**1.43**
FP16 + EDR IB	8	15	171.5	1128	**1.58**

Related Work

- Performance modeling for parameter-server based DL system on CPU cluster [Yan et al, SIGKDD, 2015]
  - The authors proposed performance model and optimizer to minimize DNN training time for CPU cluster
  - Our performance model predicts distribution of mini-batch size and staleness as well as training time
- Relation between mini-batch size, staleness and generalization error [Gupta et al, 2015]
  - The authors proposed an asynchronous parallel DL system *Rudra*
  - The authors empirically showed that generalization error of trained DNN is affected by staleness as well as mini-batch size
  - Universal knowledge about relation among mini-batch size, staleness, generalization error and training time is still unclear

Conclusion and Future Work

- Conclusion
  - Our model predicts epoch time, average mini-batch size and staleness with 5%, 9%, 19% error in average respectively on several supercomputers
  - Our model steadily choose the fastest machine configuration that nearly meets a target mini-batch size

- Future Work
  - Improving the model to support model-parallelism and more general DNN architecture
  - Combining empirical training results for more advanced prediction