Allocation of Computing Resources for Multiple Distributed Deep Learning Tasks

Liang Wei　　**Kazuyuki Shudo**

Tokyo Institute of Technology

Background: Deep Neural Network (DNN)

- To extract patterns from large datasets
  - To increase accuracy, a very deep network model with many layers is needed
  - Computation is very time consuming

AlphaGo

Acceleration of DNN training

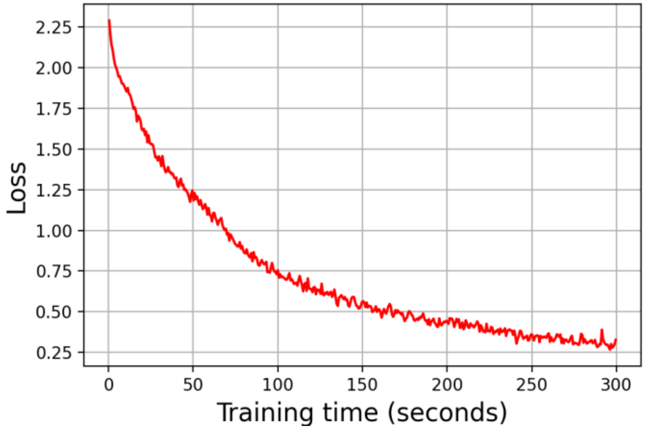
- Parallel Training
  - Train a DNN model on different GPUs
    - Model Parallelism
    - Data Parallelism

- Adjust the hyperparameters during training
  - Learning rate: how quickly the model is optimized in each iteration
    - lrDecay: Decay the learning rate after a certain number of epochs of training

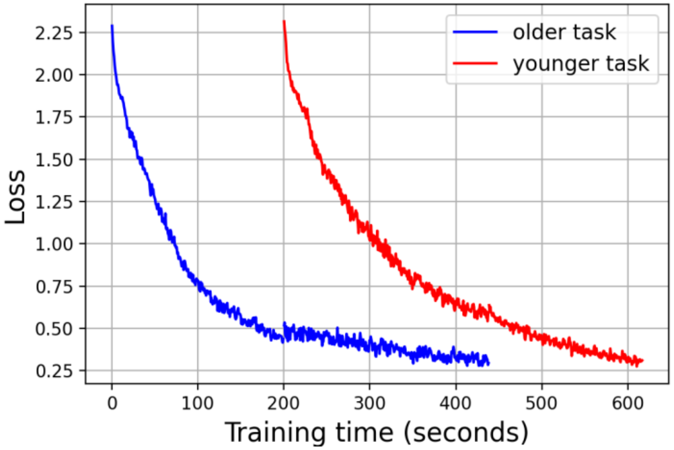
  - Batch size: how much data the model processes in each iteration
    - Dynamic Batch Size Fitting (Liu, et al. IJCNN2019): Change the batch size during different training phases.

Allocation of Computational Resources (1/2)

- A computational node with multiple training tasks running on it
- To accelerate the overall training time by dynamically adjusting the allocation of computational resources (GPU)



4 GPUs used by only one task



The older task can only use 2 GPUs after the younger one is coming.

Allocation of Computational Resources (2/2)

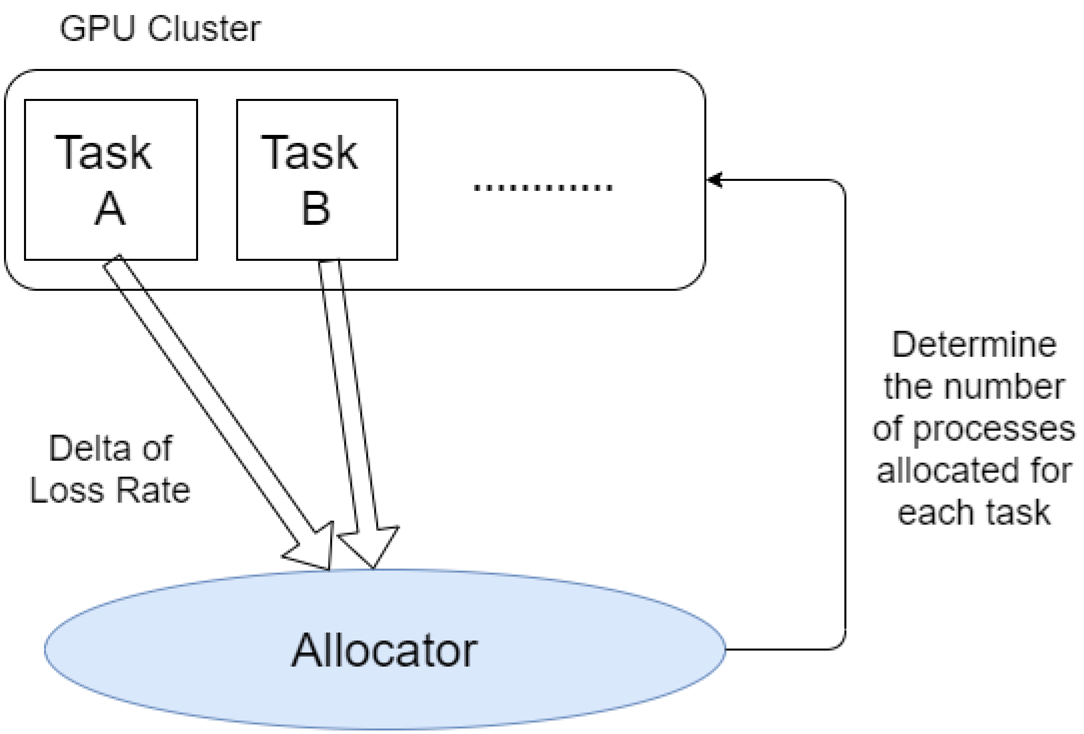
- Allocation is adjusted dynamically based on which training phase each task is in

- 4 GPUs attached to the node
  - The overall number of processes for all the training tasks will be set to 4 (※)
  - Adjustment to # of processes for each task = Adjustment to # of GPUs allocated for each task

(※) Why the overall number of processes need to match with the number of GPU?
- NCCL is used as the communication primitives in all the experiments
- It has this limitation that the number of processes cannot be more than that of GPU

Dynamic Allocation (1/2)

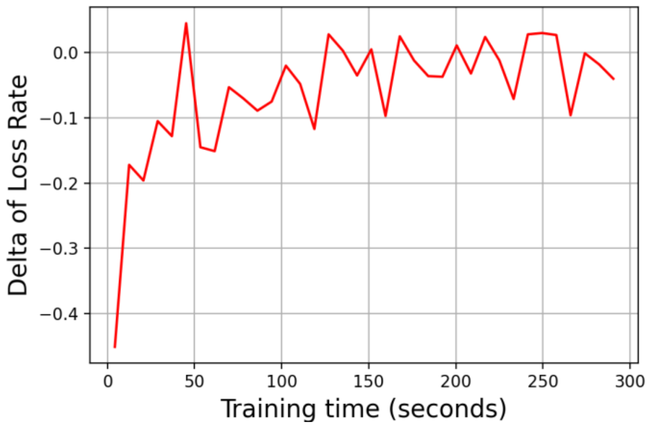
- An Allocator which has access to the training information of each task is necessary



Dynamic Allocation (2/2)

- How to determine the training phase of a task?

1. When a new task starts, run 100 iterations at first.
2. Based on the delta of loss rate, the allocator will adjust an appropriate number of processes for this task.



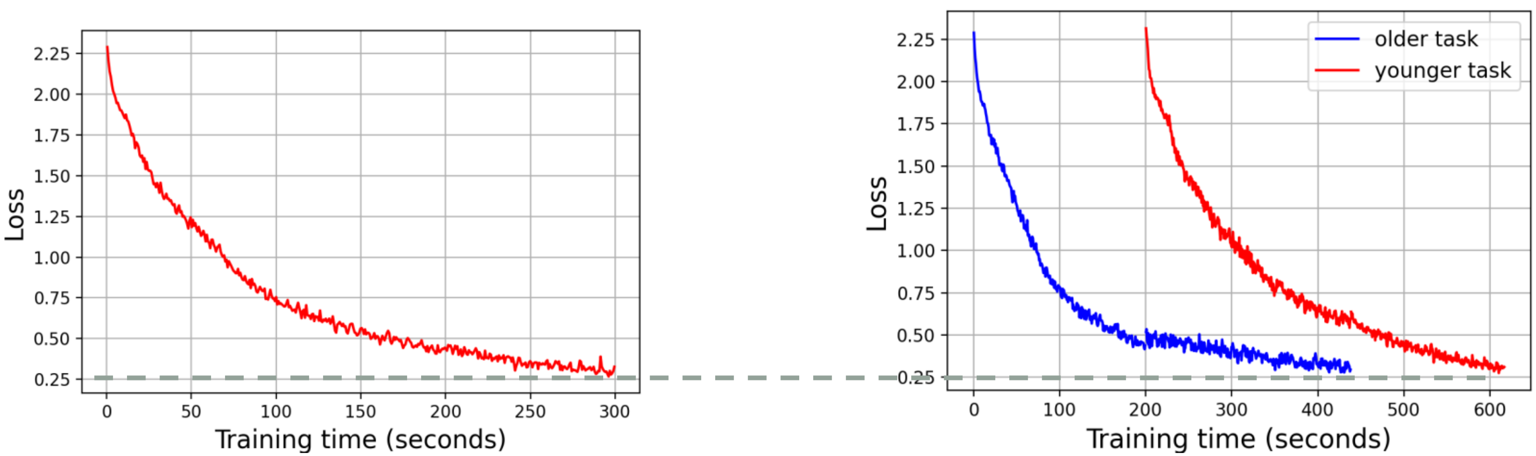
The diff of loss rate

Experimental setup

- Cluster
  - CPU: Intel Xeon CPU E5-2698
  - GPU: Tesla V100-PCIE-32GB × 4
  - OS: Ubuntu 20.04, Linux-5.8.0
- Framework: PyTorch
- Model: VGG-16、ResNet-50
- Dataset: CIFAR-10

Settings of experiments

1. The number of training tasks executing simultaneously is set to 2 (older task and younger task)
2. The time gap for the two tasks is set to 200 seconds
3. Criteria of convergence: when the loss rate reaches the level that one model has been trained for 300 seconds individually

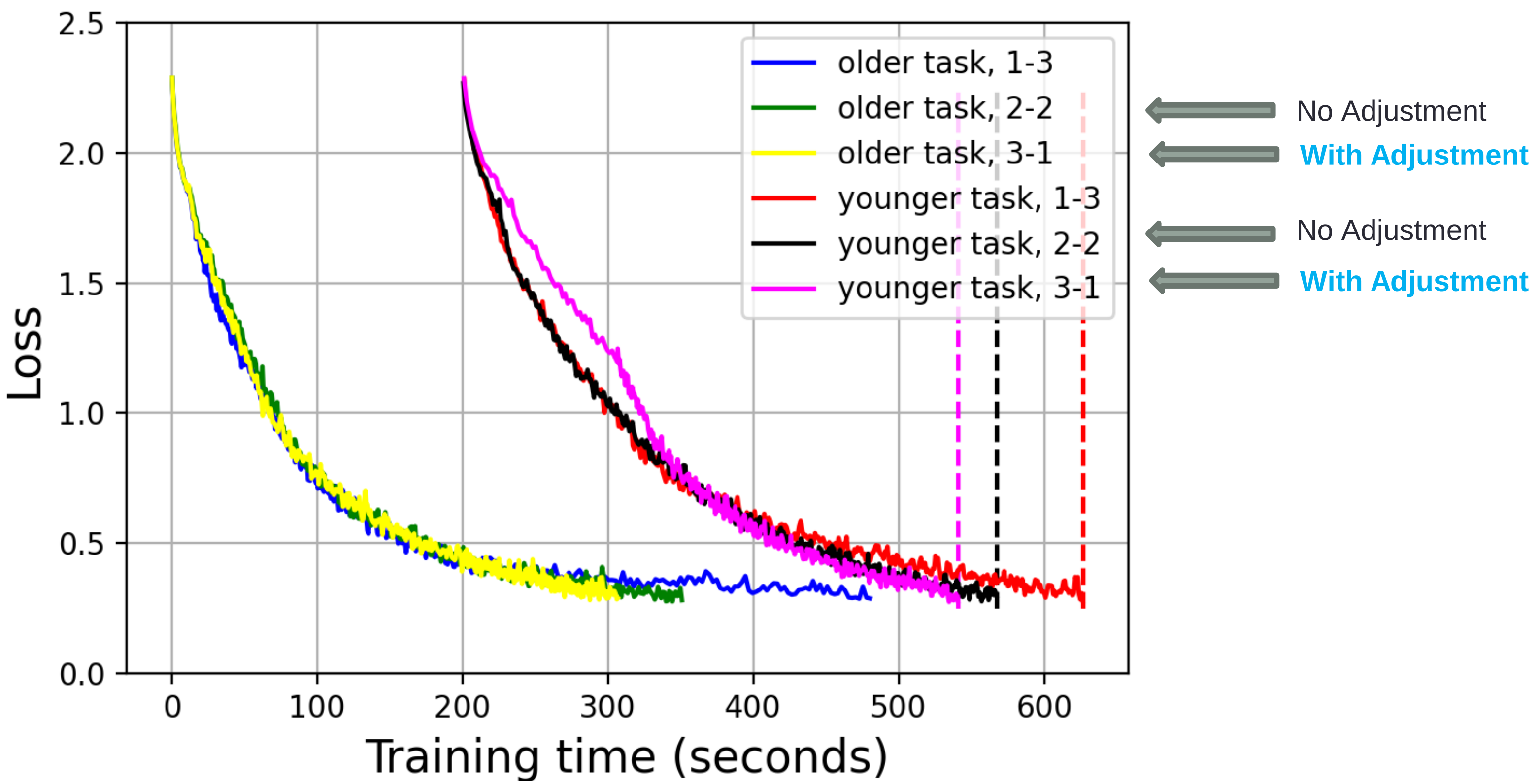


Details of experiments

- Experiment 1: Use the same model
  - DNN: VGG-16
- Experiment 2: Use two different model
  - DNN: ResNet-50, VGG-16
- Experiment 3: Training using multiple nodes
  - DNN: VGG-16
  - Nodes are connected via LAN cables.
- To not change the amount of computation, batch size for each task will be fixed to 512.

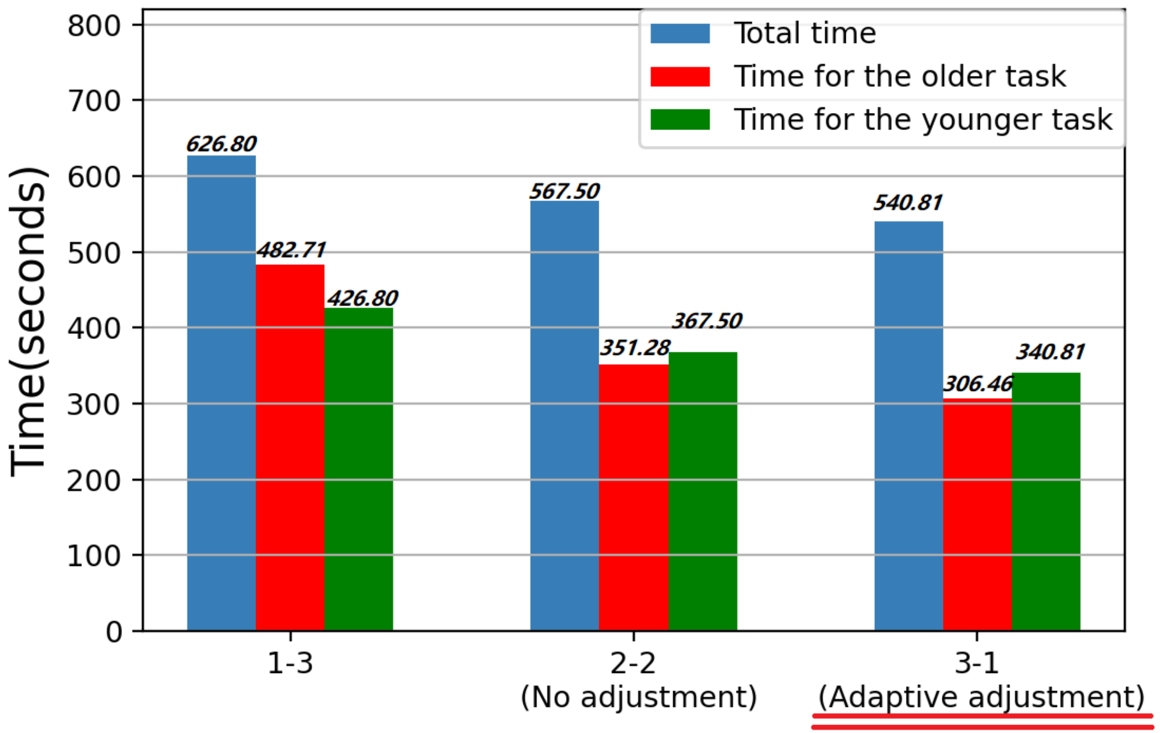
Experiment 1: Use the same model

- With dynamic allocation, total training time was shortened by 4.70%
- Older task: shortened by 12.75%



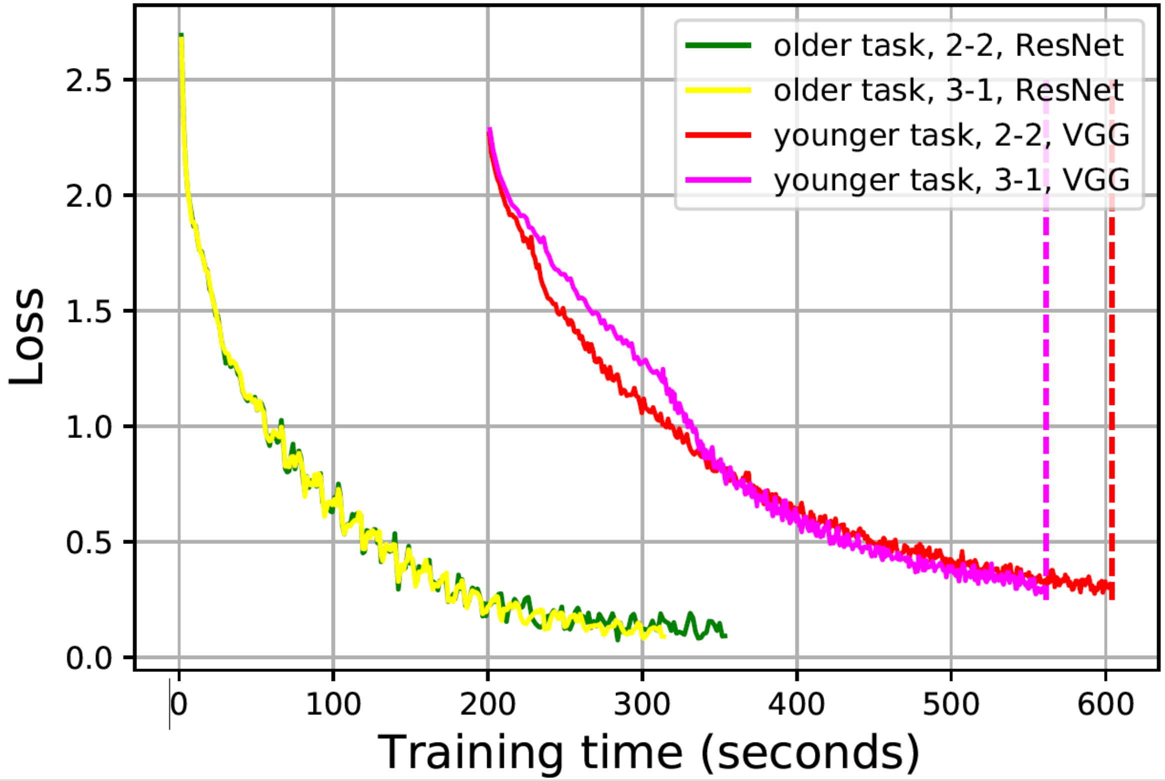
Experiment 1: Use the same model

- With dynamic allocation, total training time was shortened by 4.70%
- Older task: shortened by 12.75%



Experiment 2: Use two different models

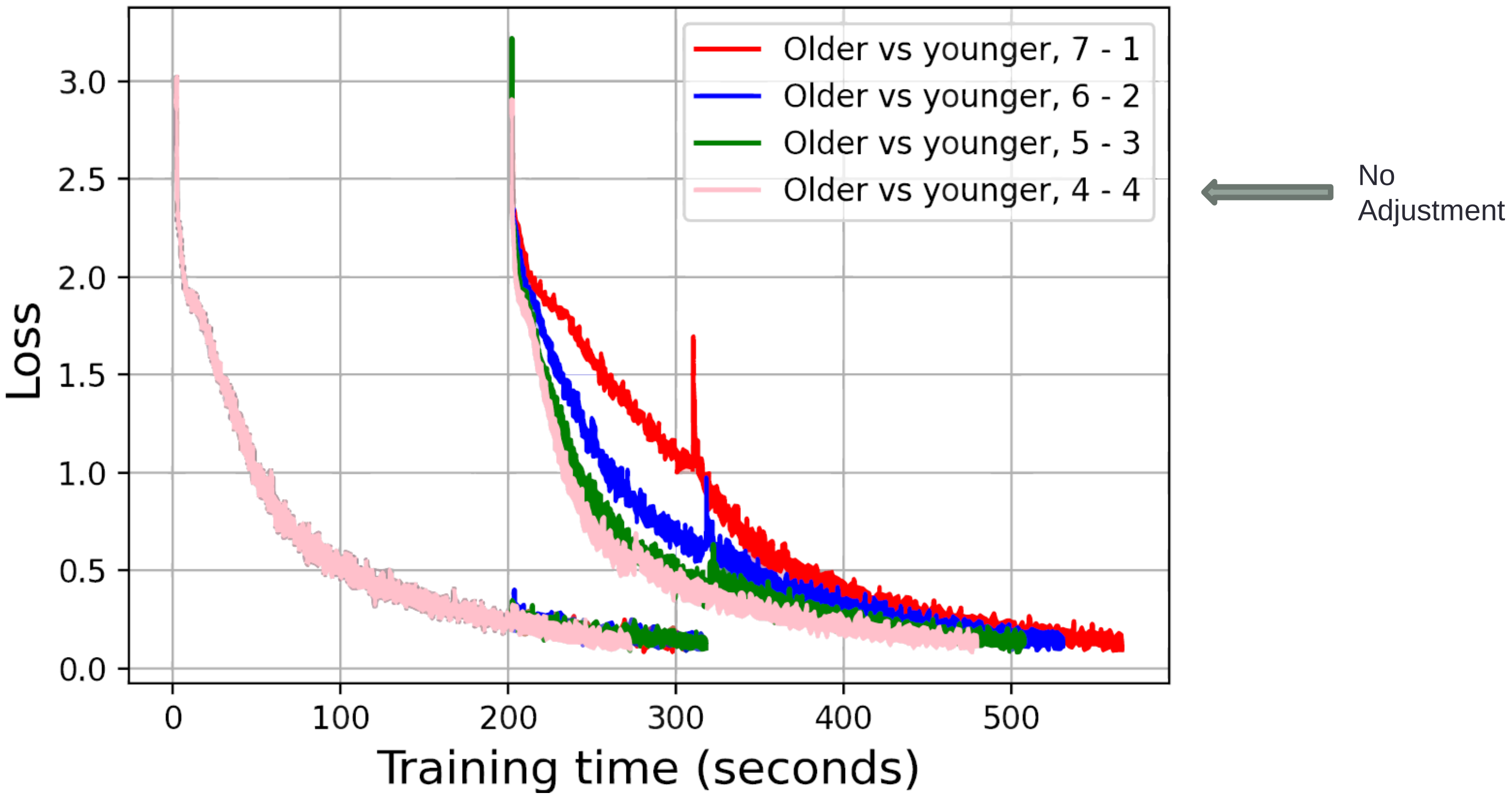
- The older task and younger task will use ResNet-50 and VGG-16 model, separately
- Total training time is shortened by 7.11%, with the older task shortened by 11.20%



No Adjustment
With Adjustment
No Adjustment
With Adjustment

Experiment 3: Training using multiple nodes

- A cluster with 2 nodes is used with 8 GPUs
- Criteria of convergence: when the loss rate reaches the level that one model has been trained for 300 seconds individually using 8 GPUs
- The result is optimal in the scenario without adjustment
  - Reason: overhead of network communication, limitation of memory …



Summary

- With dynamic adjustment based on information of training phases, the overall training time is shortened
- This adjustment is effective when using different models
- When training on multiple nodes, the training time is increased with dynamic adjustment