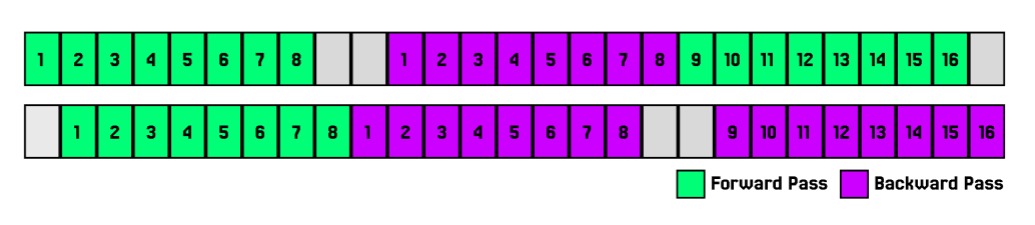
**Advantages of Pipeline Parallelism Using 2 GPUs Over Regular Training Using Single GPU**

#### ***Introduction***

This report compares the efficiency of pipeline parallelism implemented on 2 GPUs with regular training on a single GPU for a 4-layer MLP model trained on the MNIST dataset. The goal is to highlight how the pipelined approach in your code achieves improved performance by leveraging multiple GPUs.

### ***What the Code Involves***

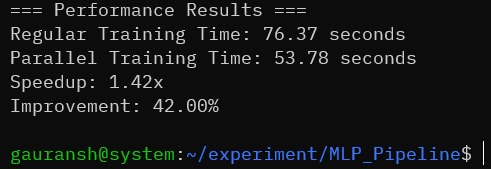
1. **Framework: PyTorch**:
   * The implementation uses **PyTorch**, a widely-used framework for deep learning.
   * Key components include:
     + **torch.nn** for defining the model architecture.
     + **torch.optim** for optimization.
     + **torch.utils.data** and **torchvision** for data loading and preprocessing.
2. **Distributed Utilities**:
   * **torch.distributed**:
     + Used to enable communication between GPUs.
     + Operations like dist.send and dist.recv manage inter-GPU data transfer.
   * **Communication Backend**:
     + NCCL for GPU-to-GPU communication, ensuring low latency.
     + Gloo as a fallback for environments without NCCL.
3. **Dataset and Preprocessing**:
   * **MNIST Dataset**:
     + Standard dataset for digit recognition.
     + Preprocessed using normalization and batching.
   * **Micro-Batching**:
     + Mini-batches are split into smaller micro-batches for efficient pipelining.
4. **Model Partitioning**:
   * The model is split into two chunks:
     + **Chunk 0 (Rank 0, GPU 1)**: Processes the first two layers.
     + **Chunk 1 (Rank 1, GPU 2)**: Processes the output from Chunk 0 and completes the forward pass.



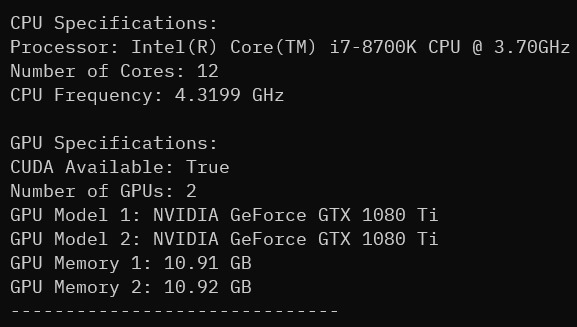
### **Advantages of Pipeline Parallelism Over Regular Training**

1. **Parallel Computation**:
   * Regular training processes all computations sequentially on one GPU.
   * Pipeline parallelism divides the model across GPUs, enabling them to compute simultaneously, reducing the overall training time.
2. **Memory Efficiency**:
   * The model's memory footprint is distributed across GPUs.
   * This allows for training larger models or increasing the batch size without exceeding GPU memory limits.
3. **Improved Throughput**:
   * Processing micro-batches in a pipelined manner ensures continuous utilization of GPU resources. Because the processing is staggered and continuous, the overall throughput (samples processed per second) increases, even though each micro-batch is smaller.
4. **Reduced Training Time**:
   * By leveraging multiple GPUs, the pipelined approach achieves significant reductions in training time compared to single GPU training.

***Speedup Observation***

******

***System Specifications***

******

***References***

1. ***Research Paper:***
   * ***Huang, Y., et al. (2019). GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism.***
2. ***Lecture:***
   * ***Carnegie Mellon University. (n.d.). Lecture 25: Parallel Deep Learning - Model & Pipeline Parallelism.***