Logistics

- Project 2 due on Wednesday

Lecture 11: An Introduction to Distributed Execution of NNs

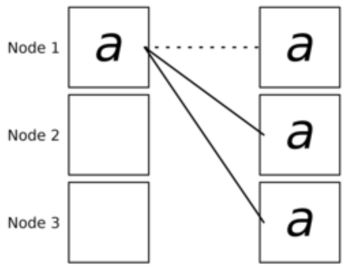
Puneet Gupta

Parallel Execution (or Training) of Large NNs

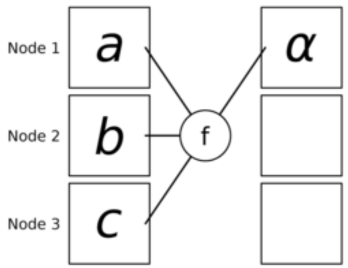
- Why do it ?
  - Model too large to fit on the DRAM on single accelerator node
    - Recall that training requires more memory with backprop: need to store *all* gradients, *all* activations and *all* weights
    - E.g., A100 GPU memory < 80GB. GPT3 inference~ 700GB memory!; GPT3 training would be 2TB+!
  - To speed up execution
    - More compute and memory BW available across multiple nodes
- How to do it ? Two basic ways:
  - Divide up the data, keep a copy of the model everywhere→ Data parallelism
  - Divide up the model, keep a copy of the data everywhere →Model parallelism
  - Can mix the two…

Collectives in Distributed ML

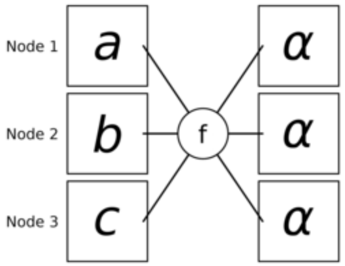
- Communication patterns between accelerator nodes
  - Several libraries to implement efficiently: Nvidia NCCL, Intel OneCCL, Blink, Xilinx ACCL, MS CCL….



Broadcast

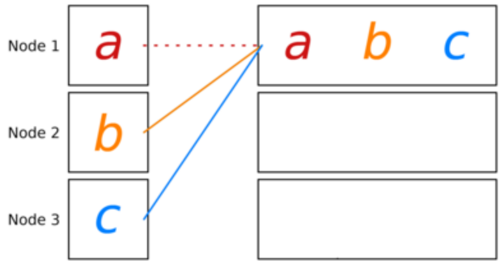


Reduce. *f* is the associative operator (e.g., sum, min, max) and α is the result of the reduction.

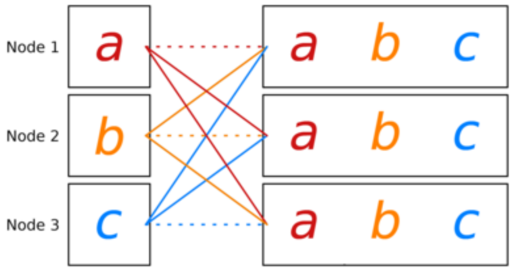


Allreduce. Result of a reduce distributed to all processing units

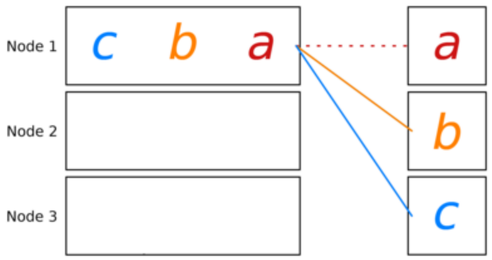
All Reduce = Reduce-Scatter + All-Gather



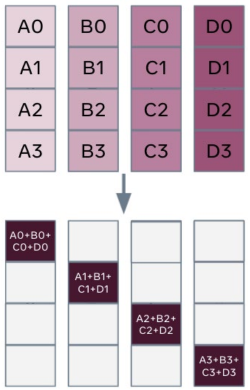
Gather



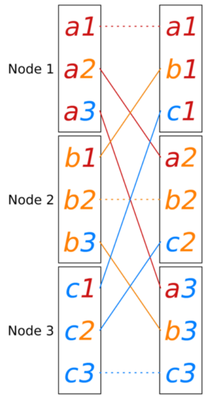
All Gather



Scatter (not same as broadcast)



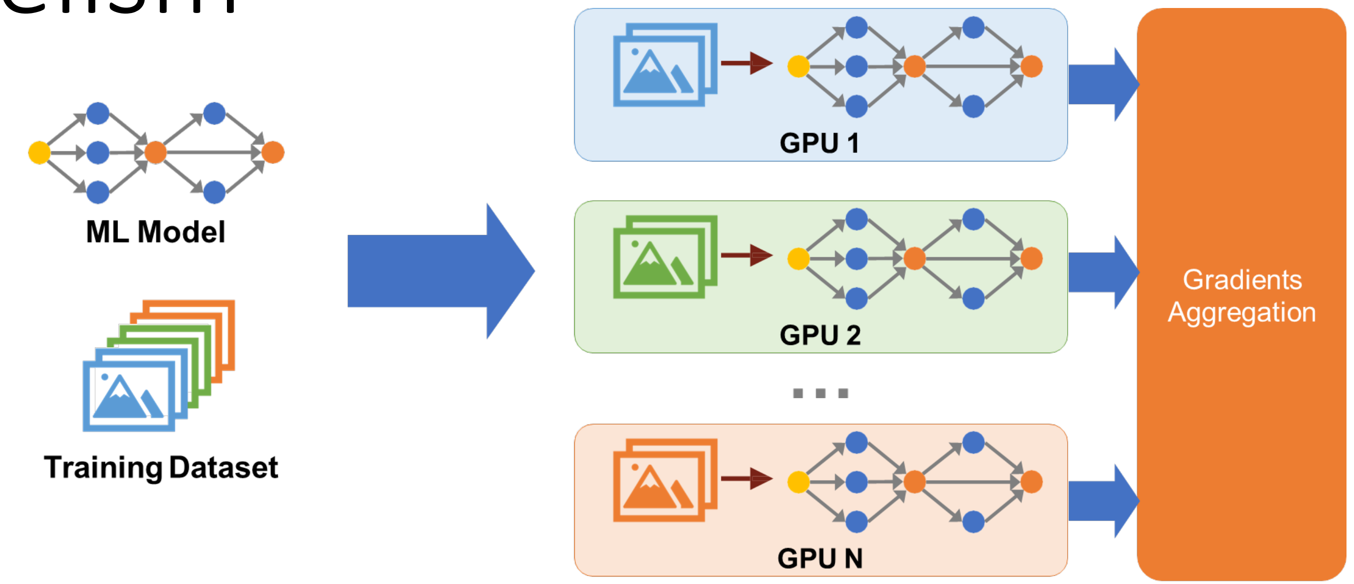
Reduce-Scatter



All-to-All

[Collective operation - Wikipedia]

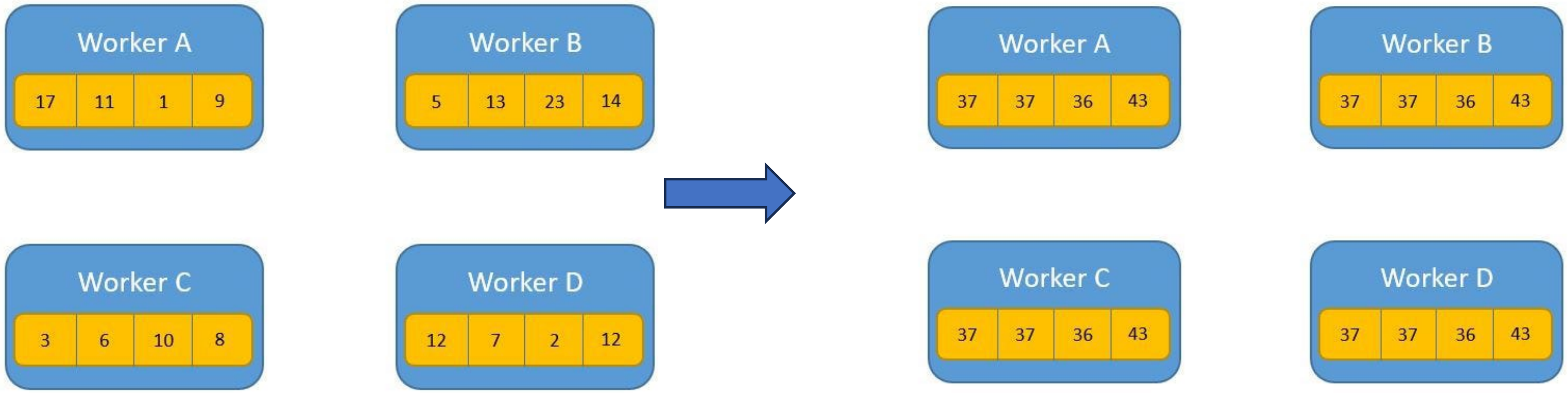
Data Parallelism



[CMU 15-418 class]

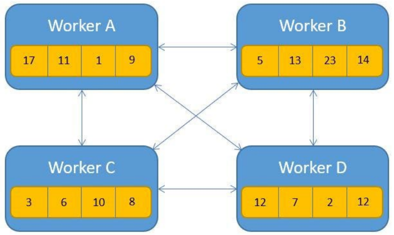
- Approach
  1. Partition training data into batches
  2. Compute the gradients of each batch on a GPU
  3. Aggregate gradients across GPUs
- Each GPU saves a replica of the entire model → Cannot train large models that exceed GPU device memory

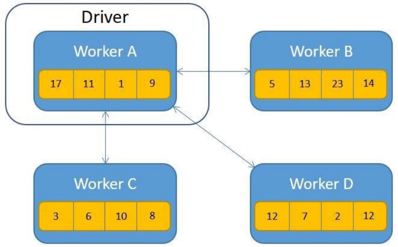
Gathering Gradients: All Reduce



[Visual intuition on ring-Allreduce for distributed Deep Learning | by Edir Garcia Lazo | Towards Data Science]

- Naieve approach: exchange gradients between every pair → 6x4x2 = 48 values over the network
- Other approaches possible
  - E.g., have a driver node → 3 x 4 x 2 = 24 values over network but one bottleneck node

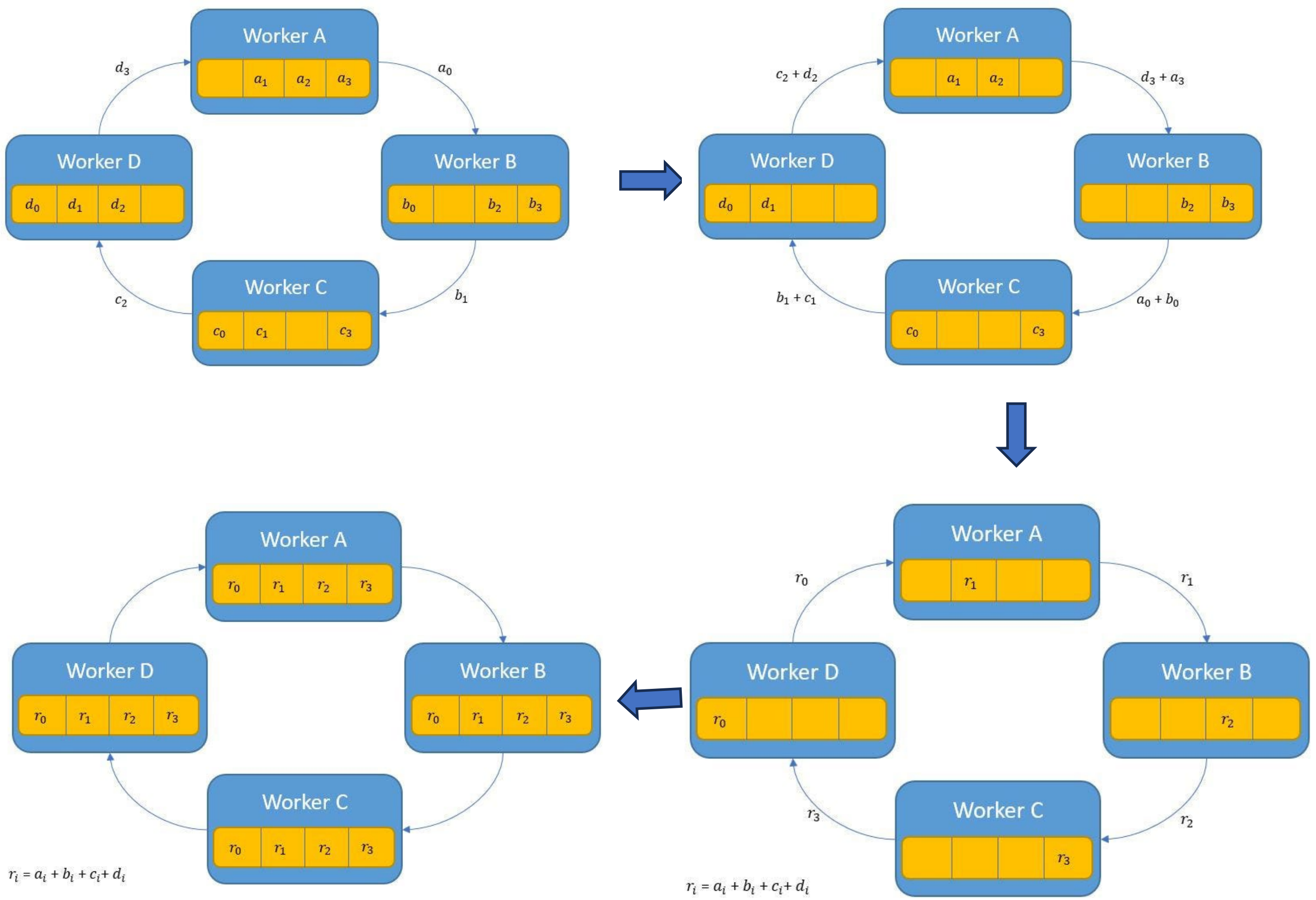




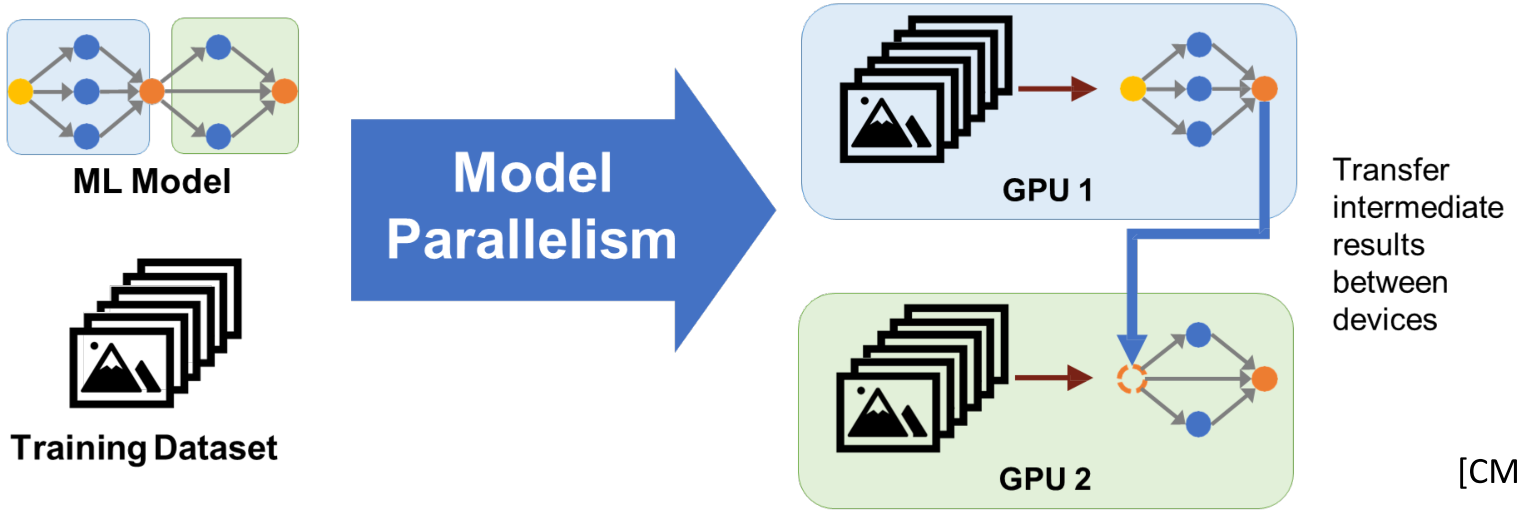
Ring All reduce

1. Share part of the gradients to ring-neighbor
2. Reduce (add) and share reduced result with neighbor
3. Repeat till fully reduced partial gradient set is there at every node
4. Share gradients without reduction in ring

- #Share-reduce cycles = #share only cycles = 4 -1 = 3;
- # values sent per node per cycle = 1
- Total values = 2 x 3 x 1 x 4 = 24



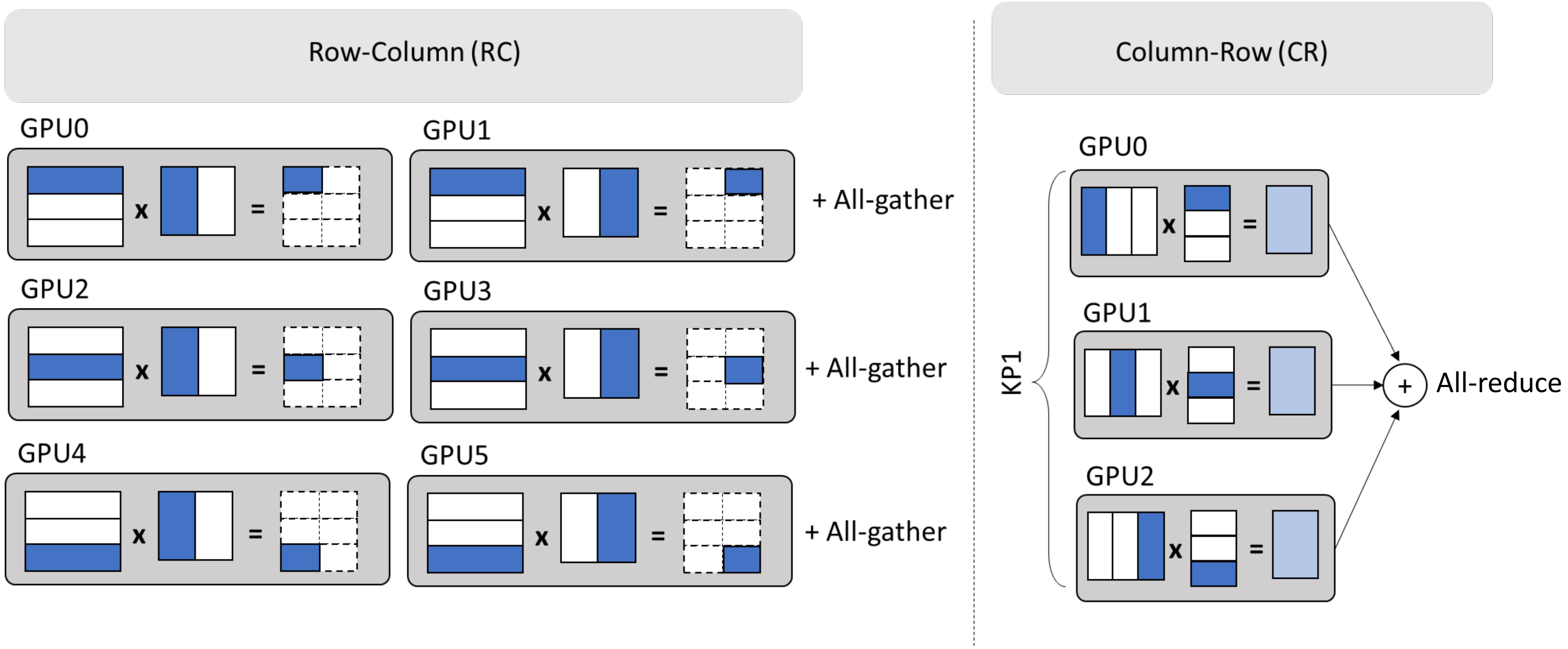
Model Parallelism



[CMU 15-418 class]

- Divide the model across machines and replicate the data.
- Supports large models and activations
- Requires communication within single evaluation
- Splitting model for balances workload across GPUs is challenging
- Data placement across multiple GPUs so that network communication latency is minimized is hard as well

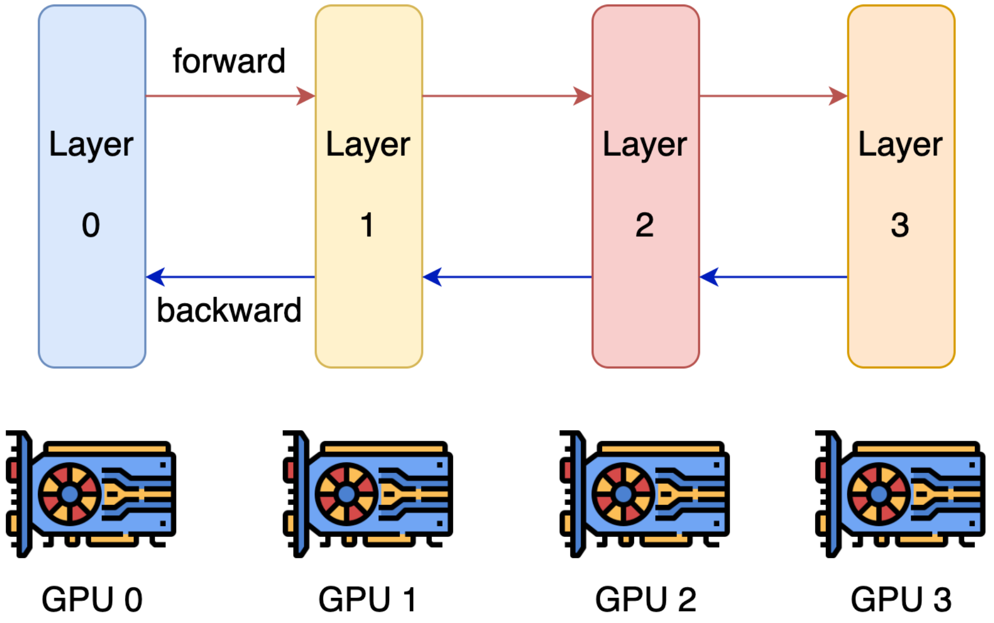
Tensor/Kernel Parallelism Strategies

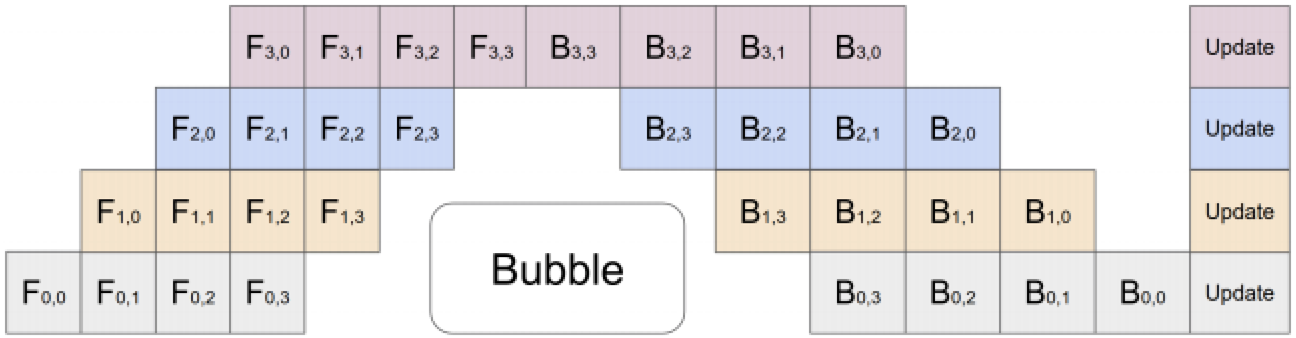


- Partition parameters/gradients within a layer

Pipeline/Layer Parallelism

- Divide model by layers. Each node takes care of one layer
  - Naively bad underutilization
    - E.g., GPU0 waits for gradients for layer 1 to flow back
  - Divide mini-batch into micro-batches and pipeline the micro-batch execution to reduce pipeline stalls.





[Paradigms of Parallelism | Colossal-AI (colossalai.org)]

Week 9 papers

- S. Teerapittayanon, B. McDanel and H. T. Kung, "Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices," *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, 2017, pp. 328-339, doi: 10.1109/ICDCS.2017.226.
- Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: generalized pipeline parallelism for DNN training. In Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP '19). Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3341301.3359646
- Yin, Shihui, et al. "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks." *IEEE Journal of Solid-State Circuits* 55.6 (2020): 1733-1743.
- Shafiee, Ali, et al. "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars." *ACM SIGARCH Computer Architecture News* 44.3 (2016): 14-26.
- Davies, Mike, et al. "Loihi: A neuromorphic manycore processor with on-chip learning." *Ieee Micro* 38.1 (2018): 82-99.