MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

HiDL
High-Performance Deep Learning

Scaling Distributed DNN Training for Large Images

CSE 5249

Tom Ballas, Zhengqi(Drago) Dong, and Arpan Jain

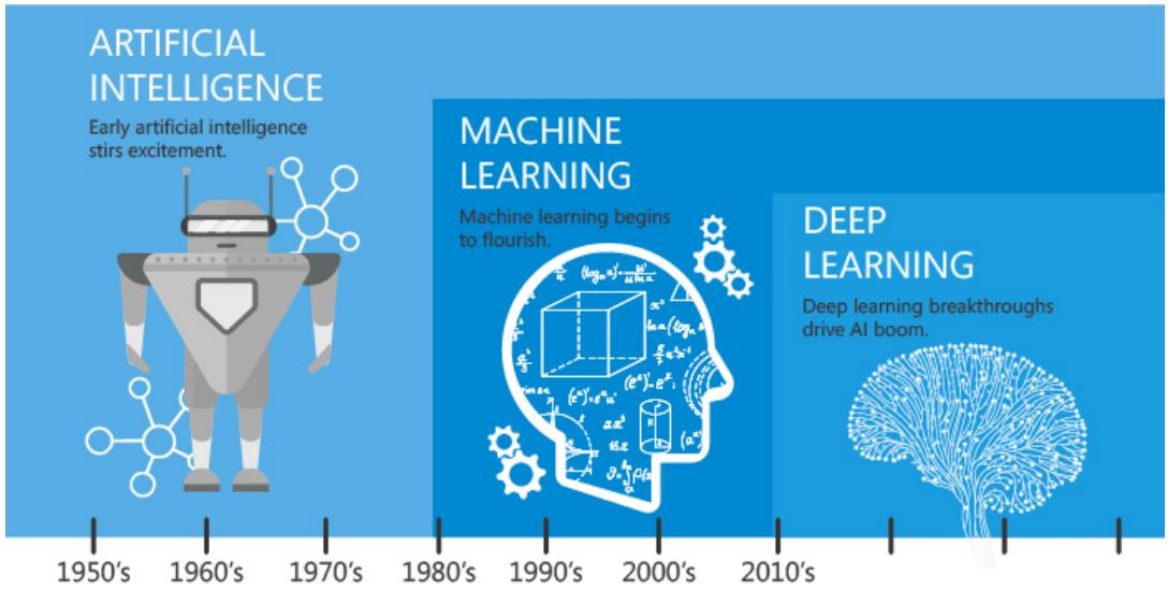
The Ohio State University

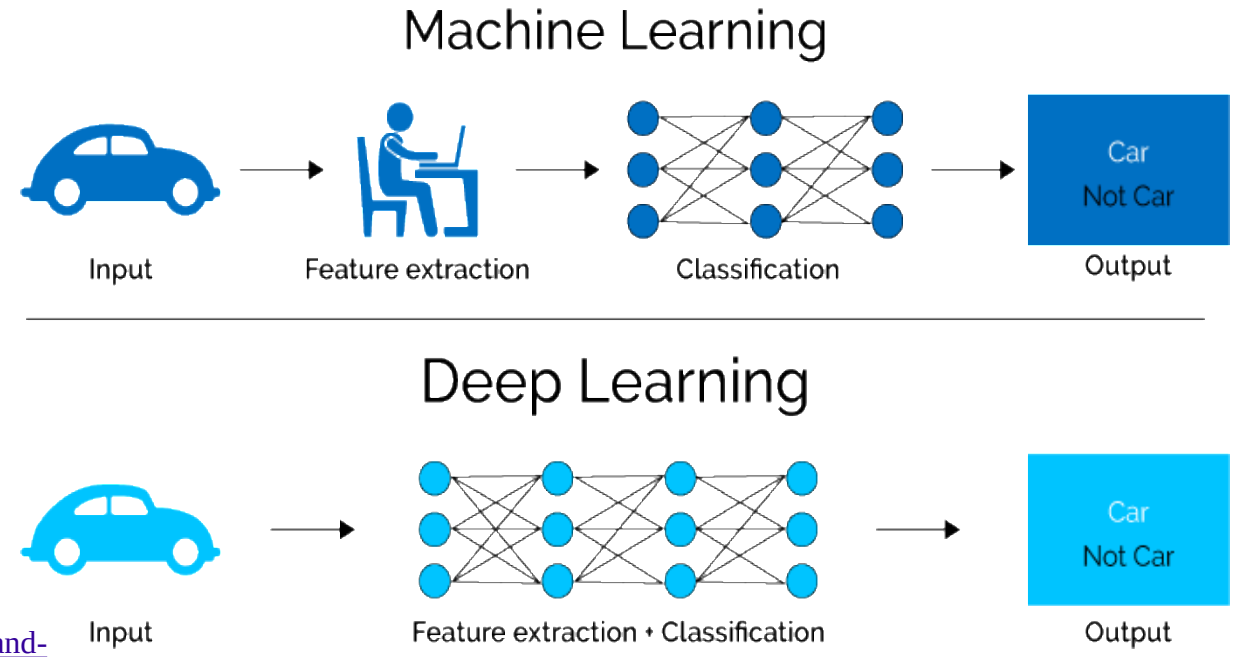
E-mail: ballas.13@osu.edu, dong.760@osu.edu, and jain.575@osu.edu,

Background

Why Deep Learning?

- **Domain:**
  - Machine Learning(ML) is a subset of AI, and DL is a subset of ML
- **AI vs ML vs DL**
  - AI is instructed based on Intelligent system
  - Machine Learning is based on statistical modeling
  - Deep Learning is based on learning data representation.
- **Why Deep Learning?**
  - Can learn(hidden pattern) from data by itself
  - Can solve more complex task, e.g., Image classification, NLP, or speech recognition.
  - Reduce human effort of feature engineering – A transformation from "Feature Engineering" to "Network Engineering"

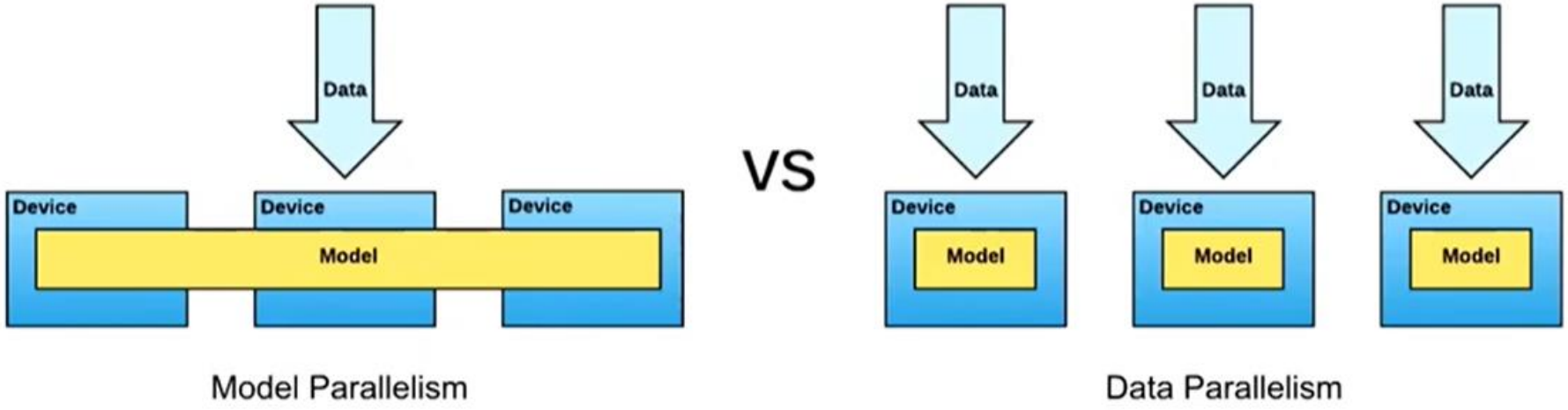




Courtesy: https://hackernoon.com/difference-between-artificial-intelligence-machine-learning-and-deep-learning-1pcv3zeg, https://blog.dataiku.com/when-and-when-not-to-use-deep-learning

Why Distributed Deep Learning?

- **The need for Parallel and Distributed Training**
  - Large scales DNNs model require a lot of memory and a lot of computation
  - Single GPU training cannot keep up with ever-larger models
  - Larger models cannot fit a GPU's memory ➔ Can be solved with Model Parallelism
- **Data Parallelism**
  - Involves replicating the entire model across multiple devices and then distributing the data across the devices
- **Model Parallelism**
  - Involves splitting the model among GPUs and use the same data for each model



Courtesy: https://www.youtube.com/watch?v=4y0TDK3KoCA, Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).

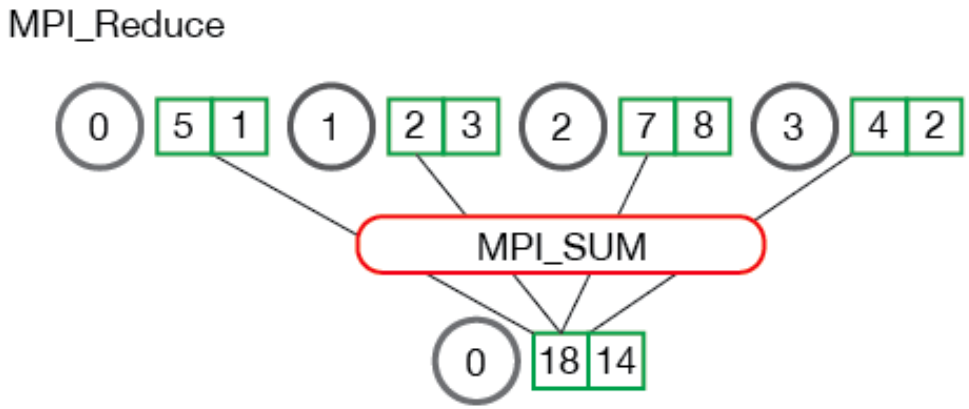
Methodologies: MPI(Message Passing Interface)

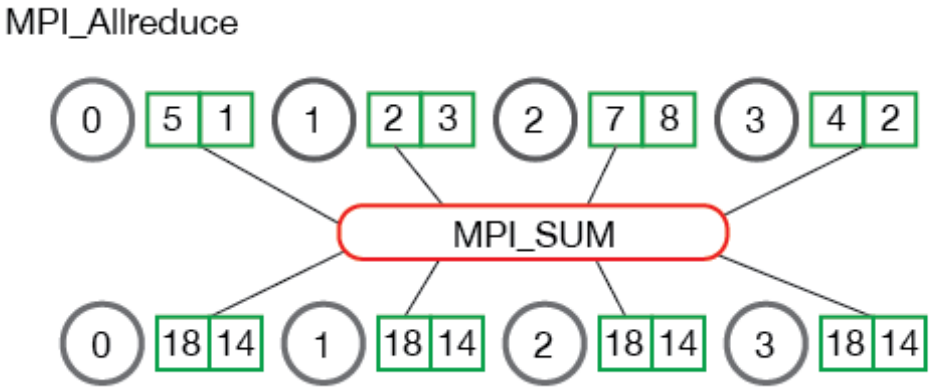
MPI Operations

- **Concept:**
  - Size: e.g. comm.Get_size()
  - Rank: e.g., comm.Get_rank()
  - Blocking/non-blocking operations
- **Point-to-point Communication**
  - **Send**: e.g. comm.send(obj, dest, tag=0)
  - **Recv:** e.g. comm.recv(source=MPI.ANY_SOURCE, tag=MPI_ANY_TAG, status=None)
- **Collective Communication**
  - Barrier: e.g., comm.barrier()
  - Broadcast: e.g. comm.bcast(obj, root=0)
  - Scatter: e.g. comm.scatter(sendobj, root=0)
  - Gather: e.g., comm.gather(sendobj, root=0)
  - All Gather: e.g. comm.allgather(sendobj)
  - All to All: e.g. comm.alltoall(sendobj)
  - **Reduce**: e.g. comm.reduce(sendobj, op=MPI.SUM, root=0)
  - **Allreduce**: e.g. comm.allreduce(sendobj, op=MPI.SUM)
  - etc…

Collective Communication: MPI Reduce and Allreduce

- **Reduce**: e.g. comm.reduce(sendobj, op=MPI.SUM, root=0)
  - involves reducing a set of numbers into a smaller set of numbers via a function.
  - It takes an array of input elements on each process and returns an array of output elements to the root process.
- **AllReduce:** comm.allreduce(sendobj, op=MPI.SUM)
  - Identical to Reduce except we don't need to specify a root process id, since the results are distributed to all processor



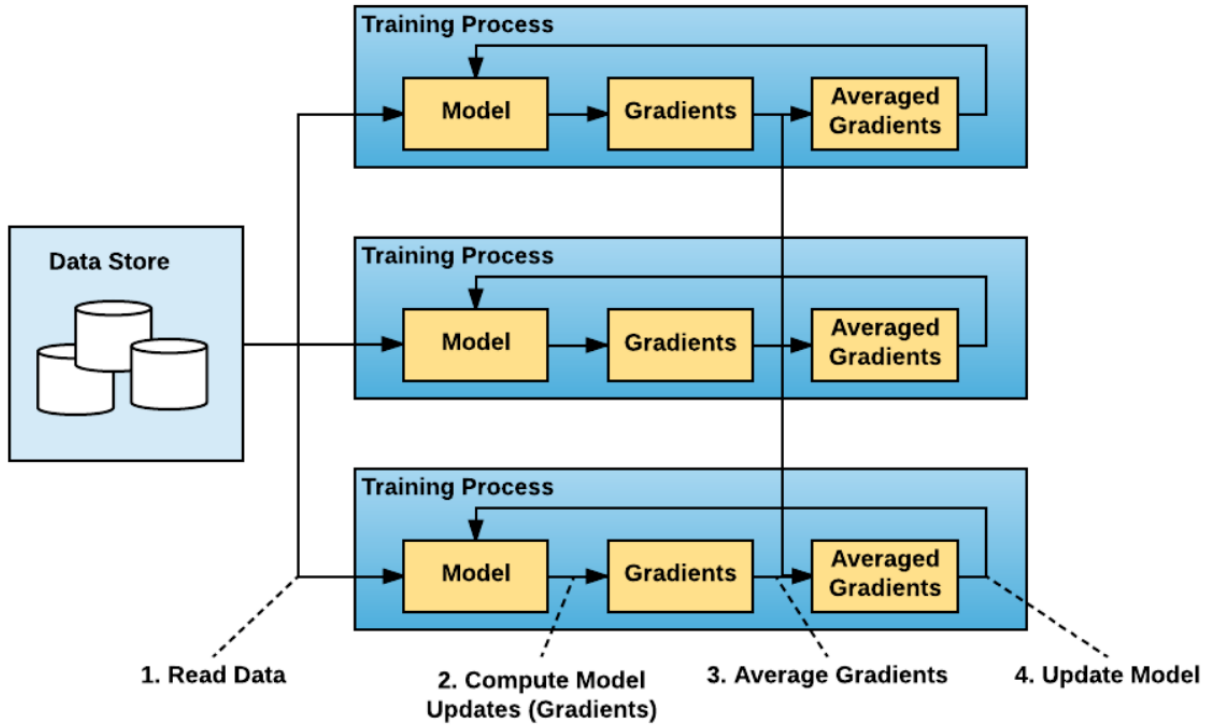


Courtesy: https://mpitutorial.com/tutorials/mpi-reduce-and-allreduce/

Methodologies: Data/Model parallelism

Data Parallelism: Definition

- **What is Data Parallelism?**
  - Involves replicating a model across multiple devices and then distributing the data across the devices
  - Each worker/machine have entire copy of model but only portion of data.
  - The weights and biases are synchronized/averaged across all devices via AllReduce operation which is provided by MPI (Message Passing Interface)
- **The trade-off of Data parallelism**
  - Allow higher parallelization and efficiencies(faster in time) than model parallelism
  - More popular and easier to implement than model parallelism
  - Cannot used for model's size larger than the memory



Courtesy: Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).

Model Parallelism: Definition

- **What is Model Parallelism**
  - The model is partitioned and distributed across multiple devices/machines, and each device works on a part of the model.
- **The need for Model parallelism:**
  - Used for model that is too large to fit a single GPU (e.g. BERT-320M, GPT-110M , GPT2-1.5B, GPT3-175B parameters)
- **Challenges for Model parallelism:**
  - Need to change the forward/backward pass into a distributed fashion across multiple GPUs, because the explicitly communication is needed between each partition.

Courtesy: http://web.cse.ohio-state.edu/~panda.2/5194/slides/5194-au20-dl-intro.pdf

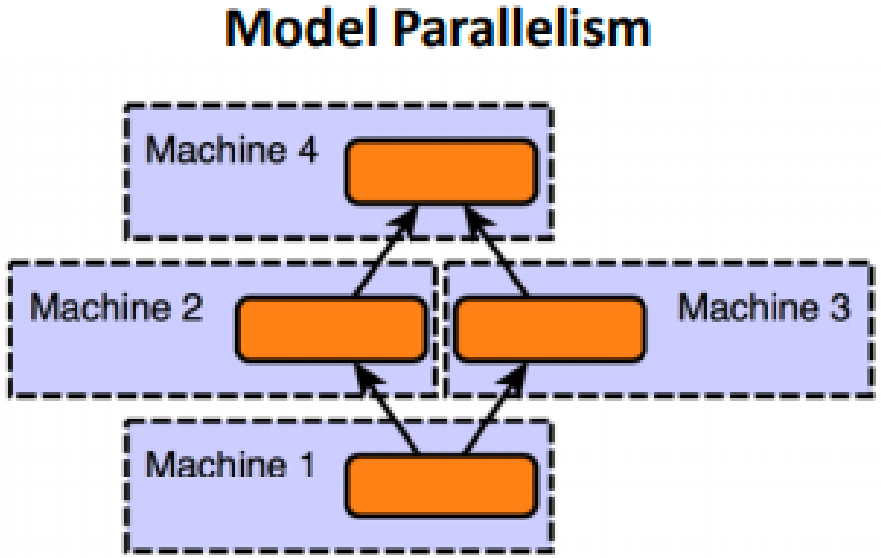


Image Size	Throughput on V100 (sec/sample)	Throughput on K80 (sec/sample)
64x64	0.0186	0.0316
128x128	0.0188	0.0533
256x256	0.0232	0.1069
512x512	0.0548	0.3005
1024x1024	0.1899	1.0024
2048x2048	0.7449	N/A
4096x4096	N/A	N/A

Note: K80 has 12 GB of memory and V100 has 32 GB. All above training used batch_size of 1.

Methodologies: U-Net Implementation

Splitting Model: ResNet

- Typically neural network allows us to split groups of layers across devices
- However, certain models present challenges in model splitting
- Example: ResNet utilizes skip connections to mitigate effects of vanishing gradient problem
  - Can split based on residual blocks instead of layers

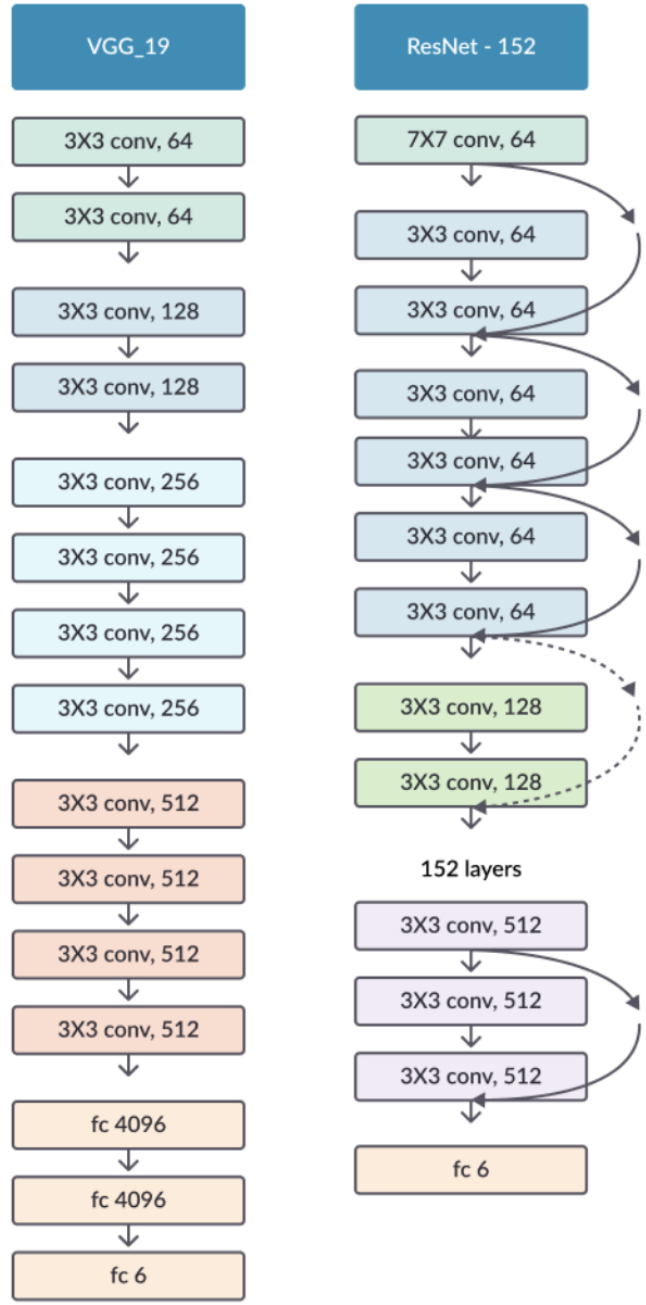


Image from: (Han et. al, 2018)

Splitting Model: UNet

- UNet architecture presents different challenge
  - The output of a layer can be the input to MULTIPLE other layers
  - Option 1: Partition Horizontally
  - Option 2: Partition Vertically

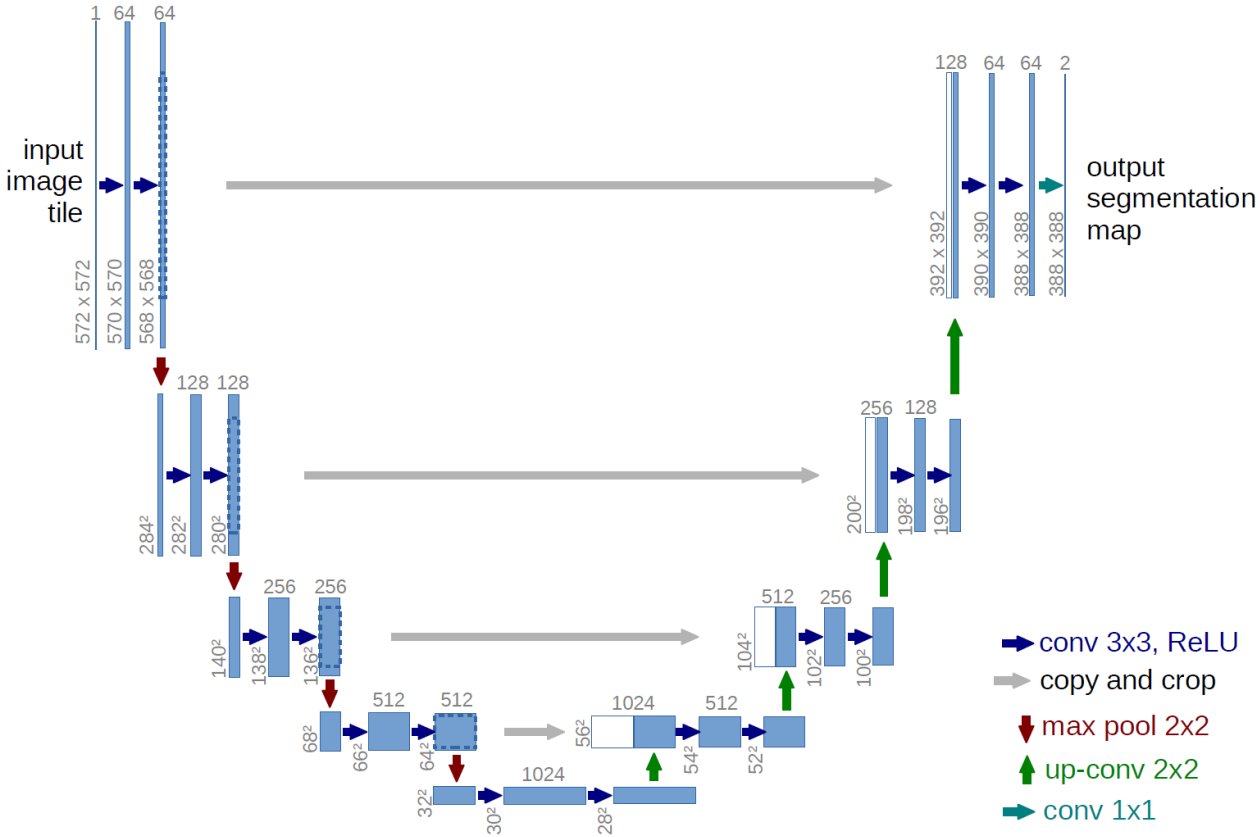
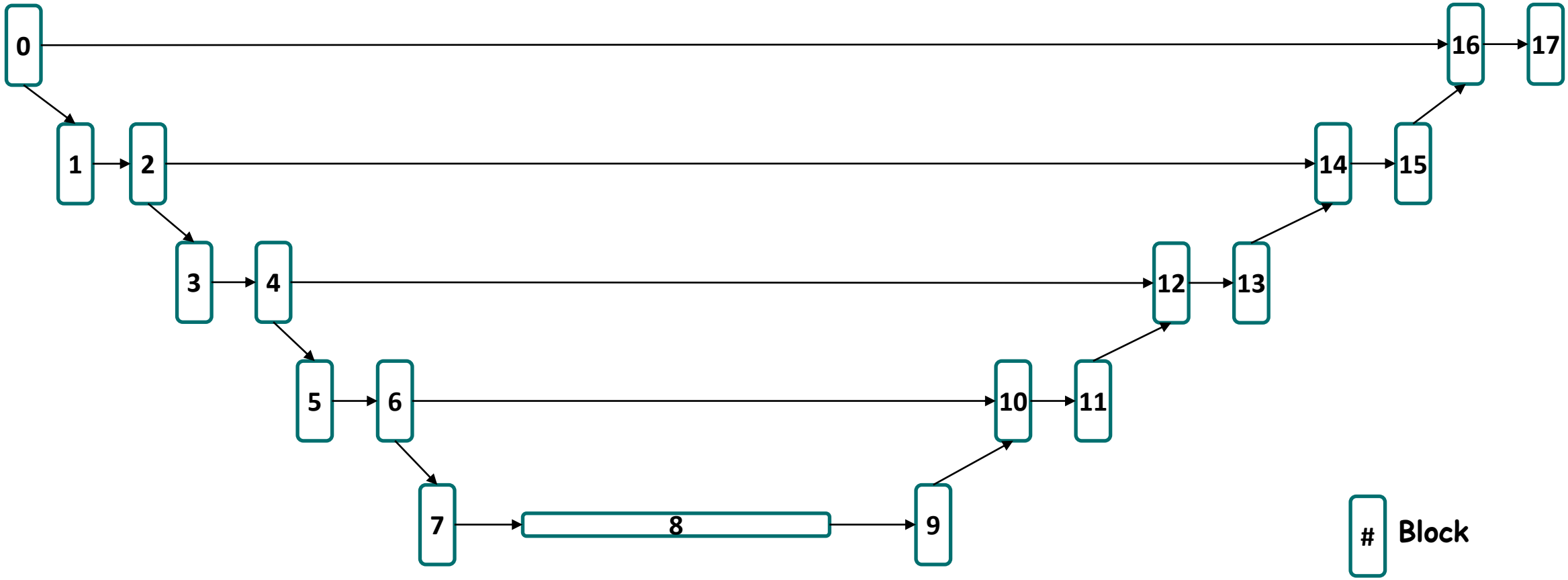


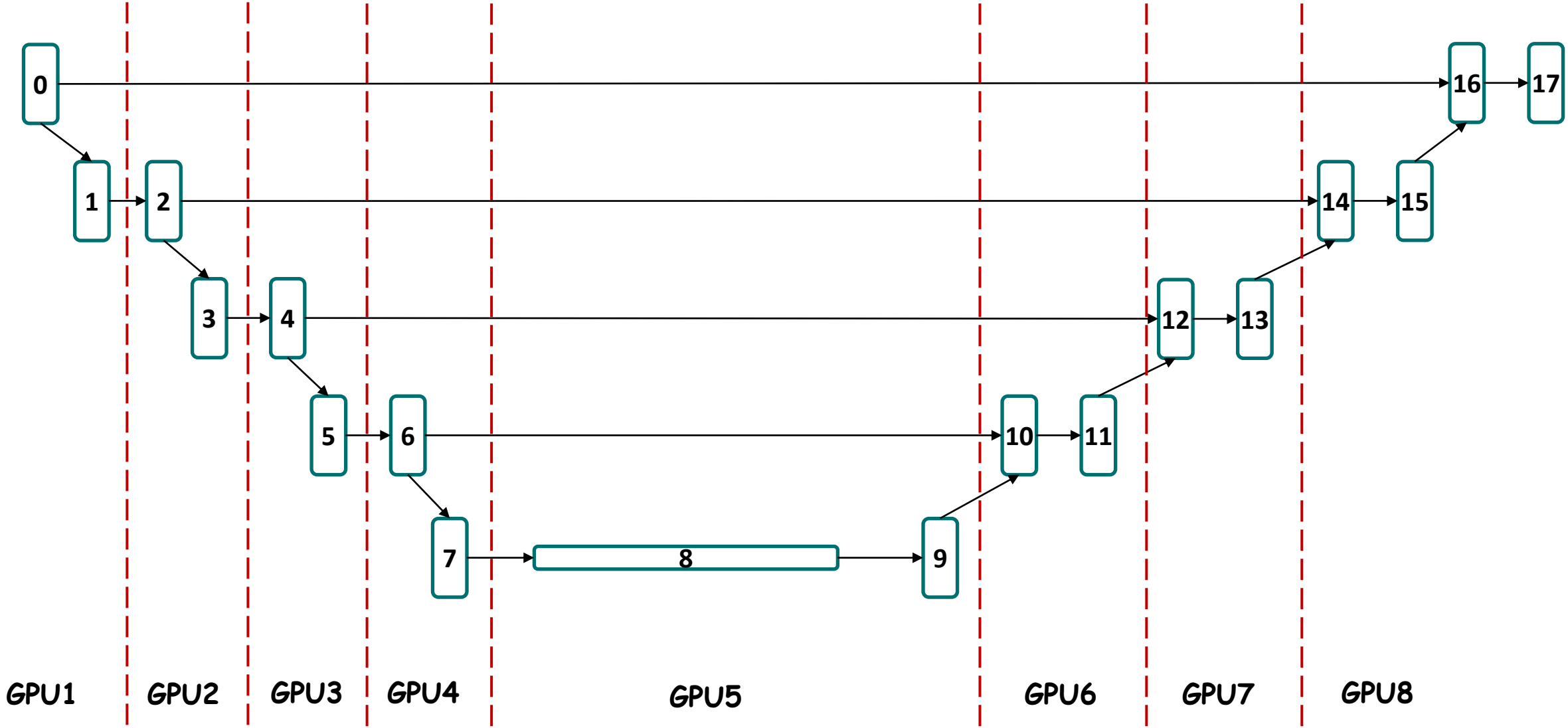
Image from: (Ronneberger et. al, 2018)

UNet (Our Implementation)

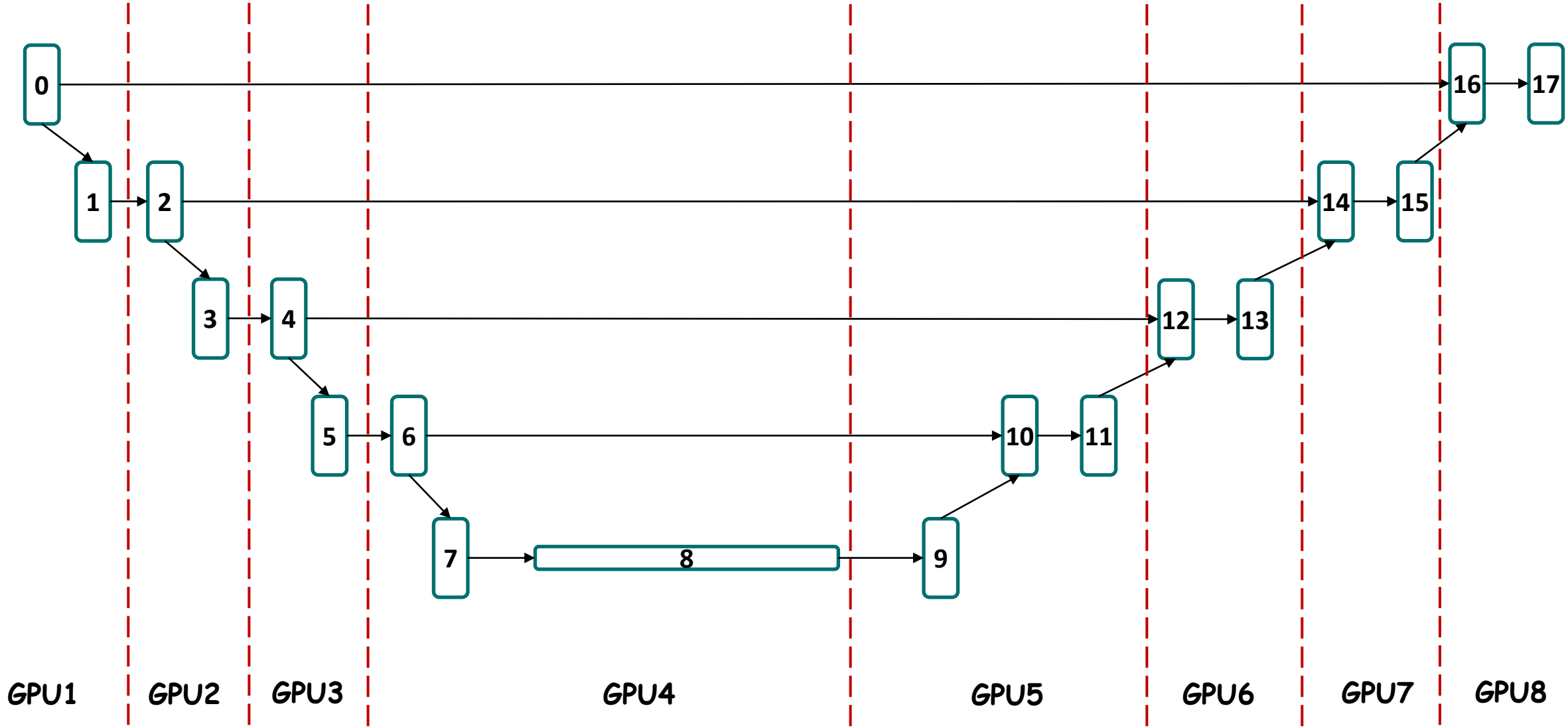
Block: Sequential model with one or more layers



Model Parallelism (Naive)



Model Parallelism (Model-Aware)



Result

V100 GPUs with 16 GB HBM

Model Parallelism on 8 GPUs

Image Size	Sequential	Model Parallelism (Naive)	Model Parallelism (Model Aware)
64	0.07481	0.19	0.19
128	0.128	0.24	0.236
256	0.346	0.5424	0.49
512	1.3	1.44	1.48
640	OOM	1.9	1.92
768	OOM	2.76	2.81
1024	OOM	OOM	5.96
2048			OOM

Conclusion

- Most of the State-of-art Deep Neural Networks are trained using Distributed DNN training
  - GPT3, T5, AmoebaNet, Megatron, Cosmoflow
- UNet has long skip connections that makes model-parallelism implementation challenging
- We showed sequential training and model-parallelism performance for a basic UNet architecture.
- Future Work
  - Try UNet with more number of layers and filters
  - Improve performance of model-parallelism using pipeline parallelism or GEMS-MASTER.

References

[1]: Vishnu, Abhinav, Charles Siegel, and Jeffrey Daily. "Distributed tensorflow with MPI." arXiv preprint arXiv:1603.02339 (2016).

[2]: Sergeev, Alexander, and Mike Del Balso. "Horovod: fast and easy distributed deep learning in TensorFlow." arXiv preprint arXiv:1802.05799 (2018).

[3]: Ammar Ahmad Awan, Arpan Jain, Quentin Anthony, Hari Subramoni, Dhabaleswar K. Panda. HyPar-Flow: Exploiting MPI and Keras for Scalable Hybrid-Parallel DNN Training with TensorFlow. in ISC HIGH PERFORMANCE 2020

[4]: Huang, Yanping, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Yonghui Wu. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." In Advances in Neural Information Processing Systems, pp. 103-112. 2019.

[5]: Arpan Jain, Ammar Ahmad Awan, Asmaa Aljuhani, Jahanzeb Hashmi, Quentin Anthony, Hari Subramoni, Dhabaleswar K. Panda, Raghu Machiraju, Anil Parwani. “GEMS: GPU Enabled Memory Aware Model Parallelism System for Distributed DNN Training.” in SuperComputing 2020.

[6]: Han, Seung & Park, Gyeong & Lim, Woohyung & Kim, Myoung & Na, Jung-Im & Park, Ilwoo & Chang, Sung. (2018). Deep neural networks show an equivalent and often superior performance to dermatologists in onychomycosis diagnosis: Automatic construction of onychomycosis datasets by region-based convolutional deep neural network. PLOS ONE. 13. e0191493. 10.1371/journal.pone.0191493.

[7]: O. Ronneberger, P.Fischer, & T. Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI) (pp. 234–241). Springer.

Thank You!

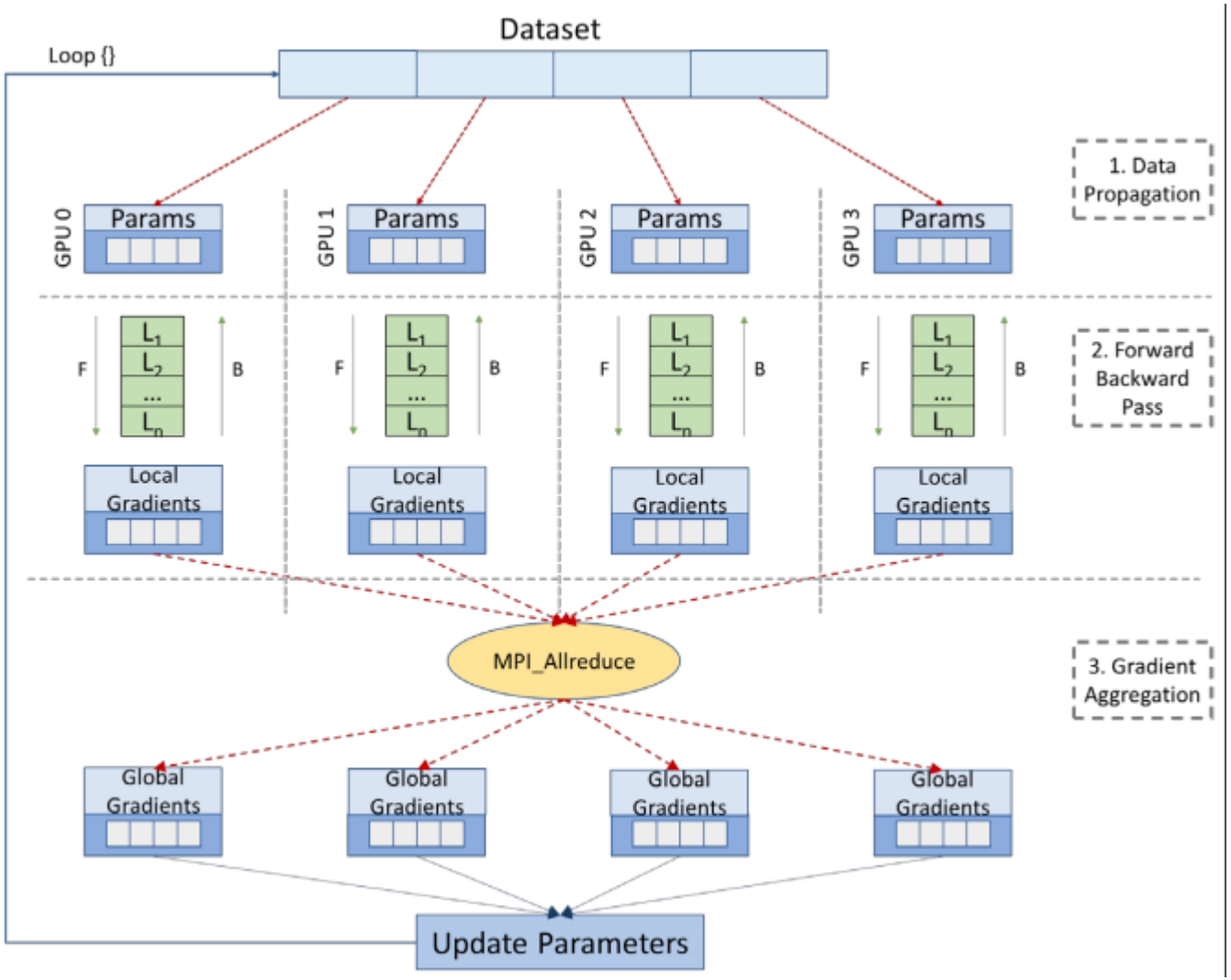
ballas.13@osu.edu

dong.760@osu.edu

jain.575@osu.edu

2. Data Parallelism: AllReduce

- Step1: Data Propagation
  - Distributed the Data among GPUs
- Step2: Forward Backward Pass
  - Perform forward pass and calculate the predicted value
  - Compute the error by comparing the prediction with ground truth label
  - Perform backward propagation and compute the gradients
- Step3: Gradient Aggregation
  - Call MPI_Allreduce to reduce the local gradients
  - Update parameters locally using global gradietns



Courtesy: http://web.cse.ohio-state.edu/~panda.2/5194/slides/5194-au20-dl-intro.pdf

3. Model Parallelism: HyPar-Flow (support Residual Connection)

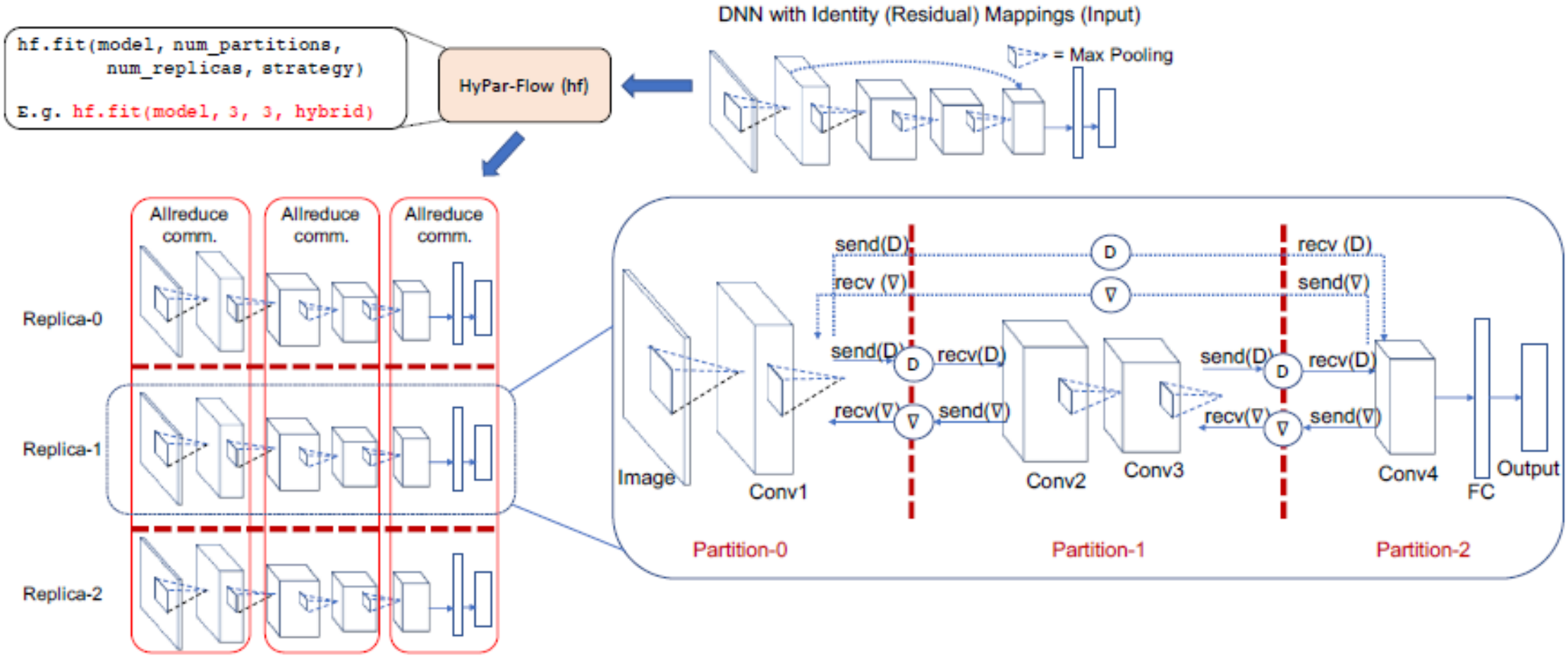
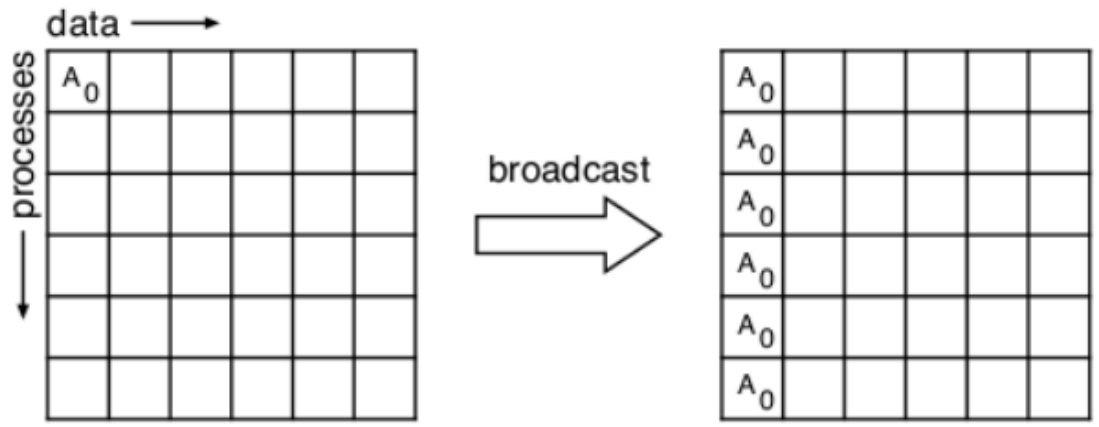


Fig. 2. Proposed User-transparent Hybrid-Parallel Training Approach (HyPar-Flow)

Courtesy: Ammar Ahmad Awan, Arpan Jain, Quentin Anthony, Hari Subramoni, Dhabaleswar K. Panda. HyPar-Flow: Exploiting MPI and Keras for Scalable Hybrid-Parallel DNN Training with TensorFlow. in ISC HIGH PERFORMANCE 2020

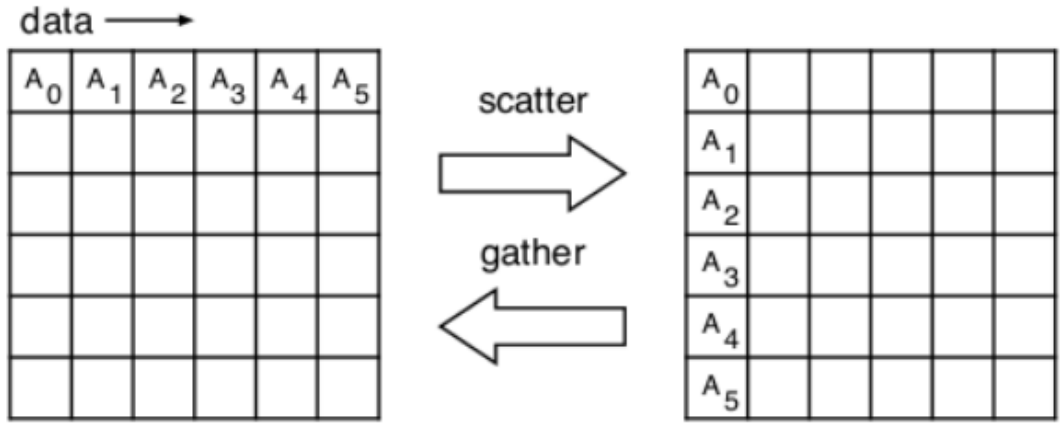
Collective Communication Operations:

comm.bcast(obj, root=0)

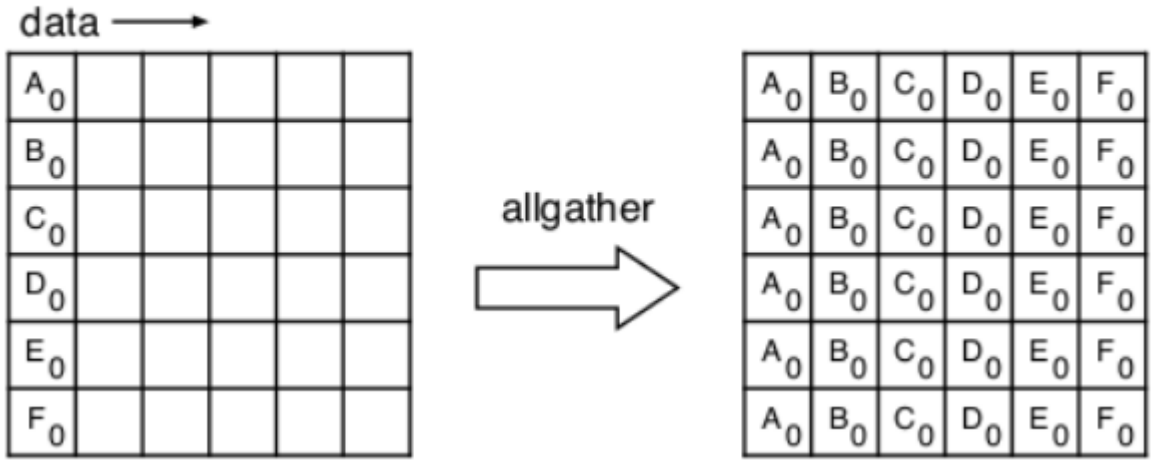


comm.scatter(sendobj, root=0) - where sendobj is iterable

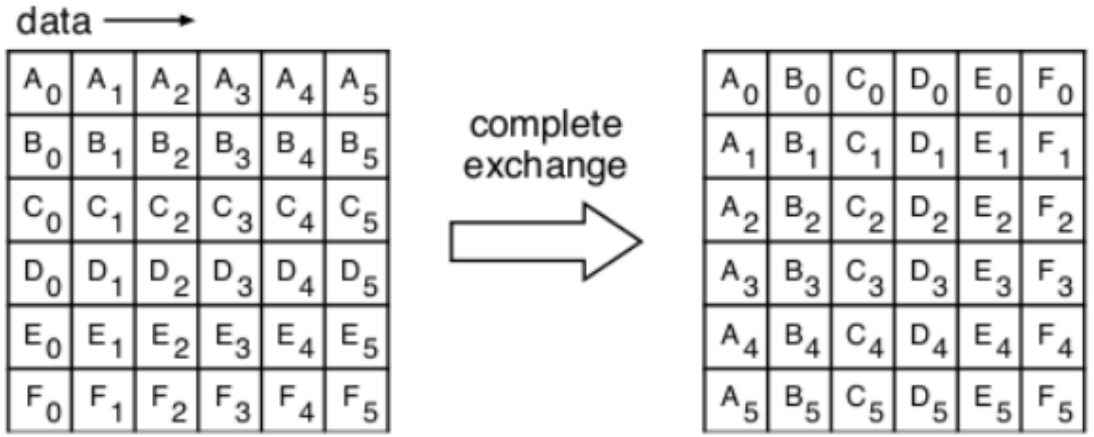
comm.gather(sendobj, root=0)



comm.allgather(sendobj) - where sendobj is iterable



comm.alltoall(sendobj) - where sendobj is iterable



Device Info: sky-k80 vs bdw-v100

```
[dong.760@gpu01 ~]$ nvidia-smi
Mon Nov 23 11:40:10 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.06    Driver Version: 450.51.06    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  Off  | 00000000:03:00.0 Off |                    0 |
| N/A   45C    P0    37W / 250W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
[dong.760@gpu01 ~]$ ^C
[dong.760@gpu01 ~]$ squeue -u dong.760
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            285299  bdw-v100       sh dong.760  R      17:57      2 gpu[01,11]
```

```
[dong.760@gpu21 ~]$ nvidia-smi
Mon Nov 23 14:55:37 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.51.06    Driver Version: 450.51.06    CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:1A:00.0 Off |                    0 |
| N/A   29C    P0    57W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K80           Off  | 00000000:1B:00.0 Off |                    0 |
| N/A   25C    P0    74W / 149W |      0MiB / 11441MiB |     98%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
WARNING: infoROM is corrupted at gpu 0000:1A:00.0
WARNING: infoROM is corrupted at gpu 0000:1B:00.0
[dong.760@gpu21 ~]$ squeue -u dong.760
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            285302   sky-k80       sh dong.760  R       1:06      2 gpu[21-22]
```