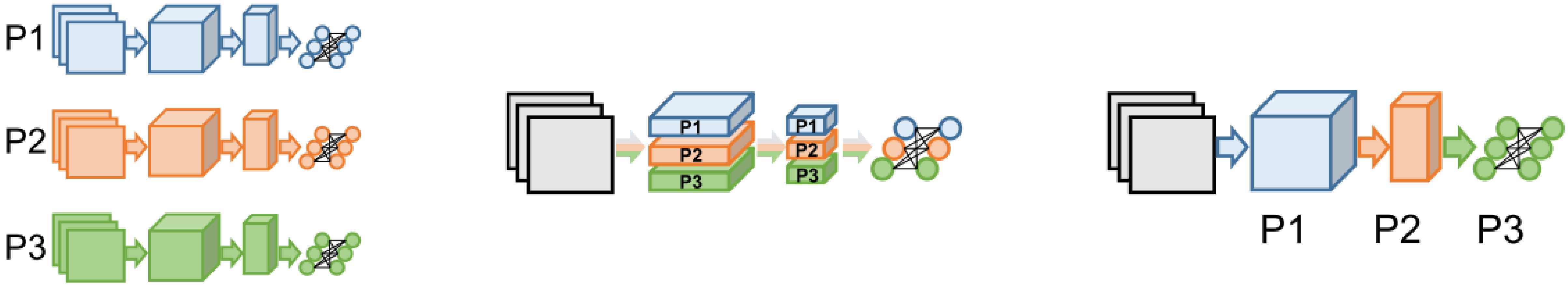
OLCF Training: 2025 Best Practices for AI on Frontier

Aristeidis Tsaris[1], Junqi Yin[1], Sajal Dash[1]

[1]Oak Ridge National Laboratory (ORNL)

Data Parallel



Data Parallelism

Distribute input samples

Model Parallelism

Distribute network structure, within or across layers

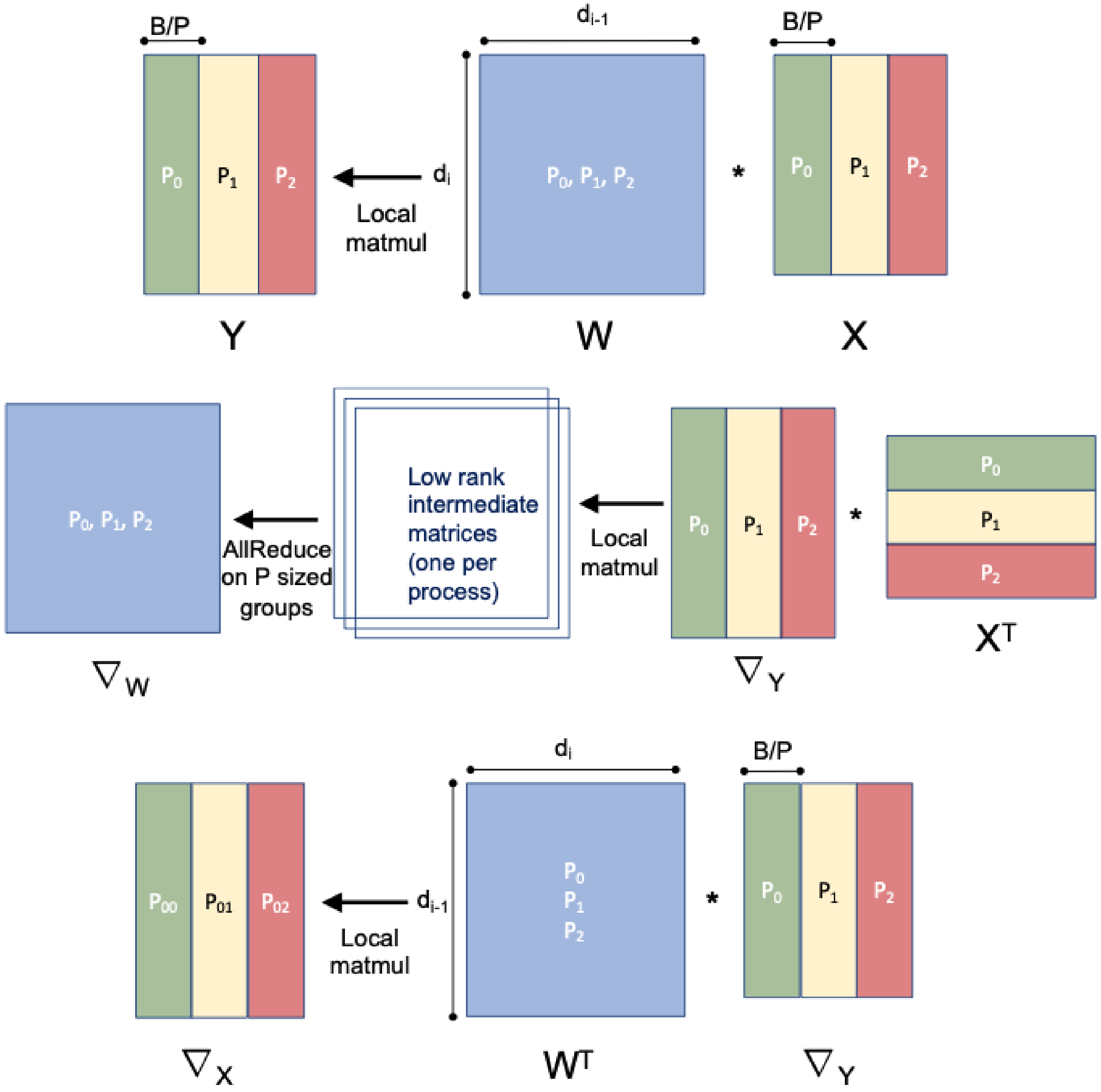
Data-parallel is still the most used

Even on large model era, data-parallel is used at scale

Fig. credit: Ben-Nun and Hoefler arXiv:1802.09941

Data Parallel

- Forward:
  - Batches are distributed across ranks
  - Weight metrices are replicated across ranks
  - Computation locally, no communication necessary
- Backward
  - AllReduce is required to sum gradients across ranks



Fig, credit by A. Buluc

Data Parallel

- Computation grows with communication; good scaling
- Global batch-size grows with the number of ranks, can negatively affect convergence (needs careful learning-rate considerations at scale)

- Performance considerations:
  - use RCCL for communication
  - tune bucket communications to reduce number of allreduce calls (DDP's bucket_cap_mb is 25MB by default; you may see benefit with larger cap)
  - When the model is small, and the data is large consider using node local NVMe burst buffer for better I/O performance

Sharded Data Parallelism (FSDP)

Standard data parallelism fully replicates model weights and optimizer states

FSDP reduces memory cost by communicate parameters only when need it

- More communication expensive that DDP, using All-Gather and Reduce-Scatter
- But simpler that model parallel approaches
- It will have a limit on model size eventually due to memory peak

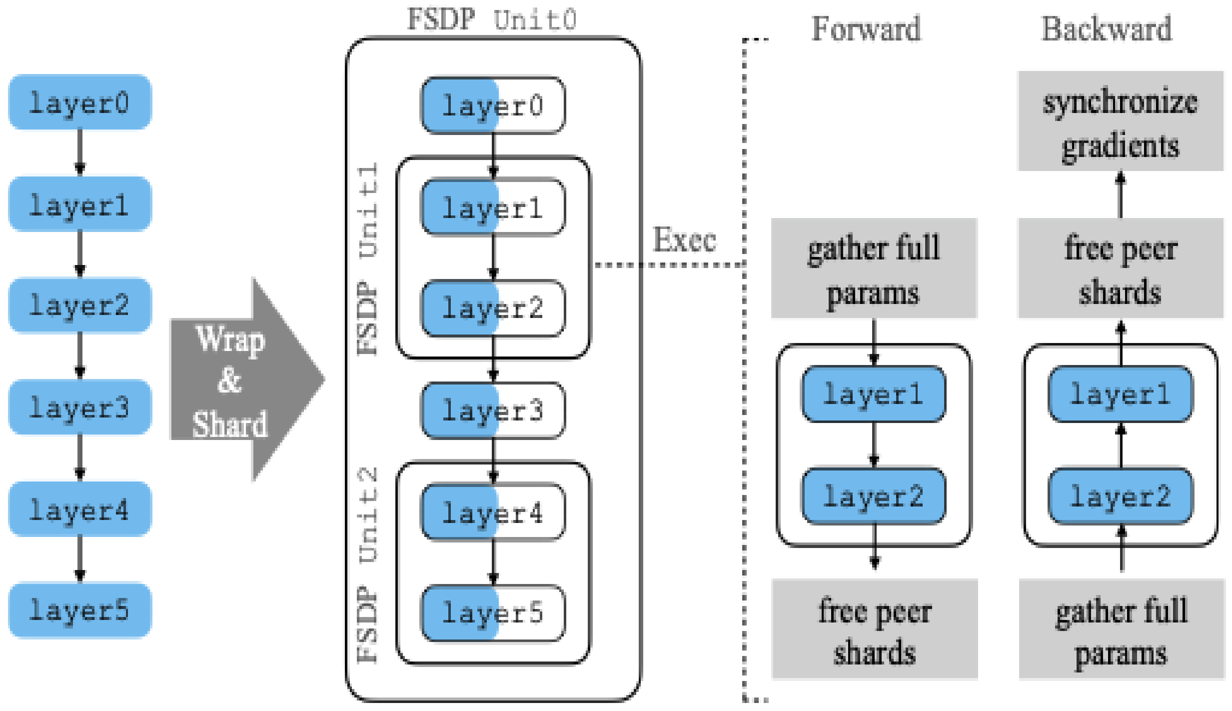


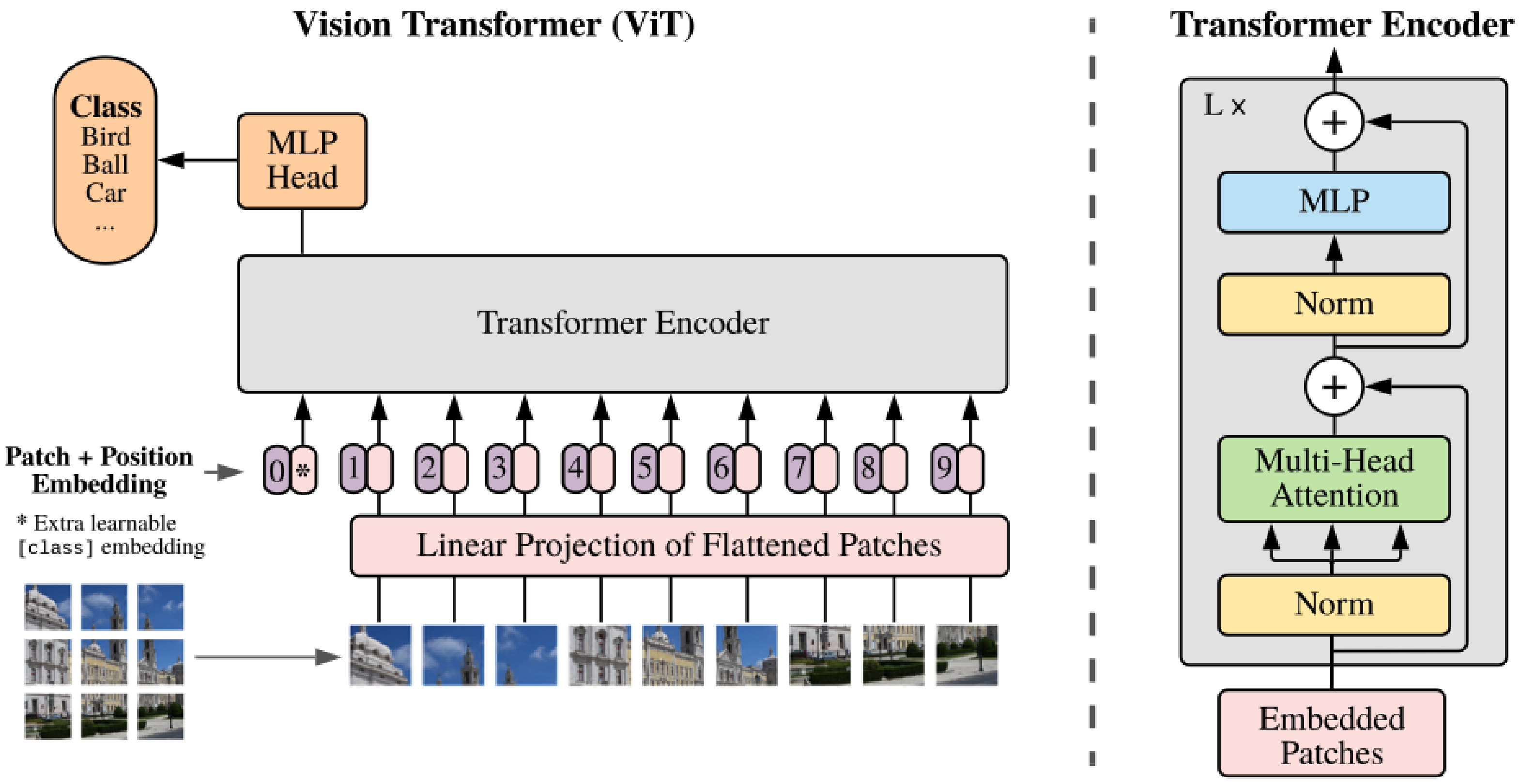
Fig. credit: https://arxiv.org/abs/2304.11277v2

Vision Transformers Introduction

Break images into patches to make tokens

Usually linear or convolutional layers are used

Decoder layers can be from MLP (simple classification) to even a whole new architecture depending on the task



Self-attention is the workhorse of ViTs:

- Input features are projected to query, key, value tensors
- Compute ‘similarity’ between queries and keys using softmax
- Multiply values by attention matrix

Fig. credit: arXiv:2010.11929

Test ViT on Frontier with FSDP for RS data

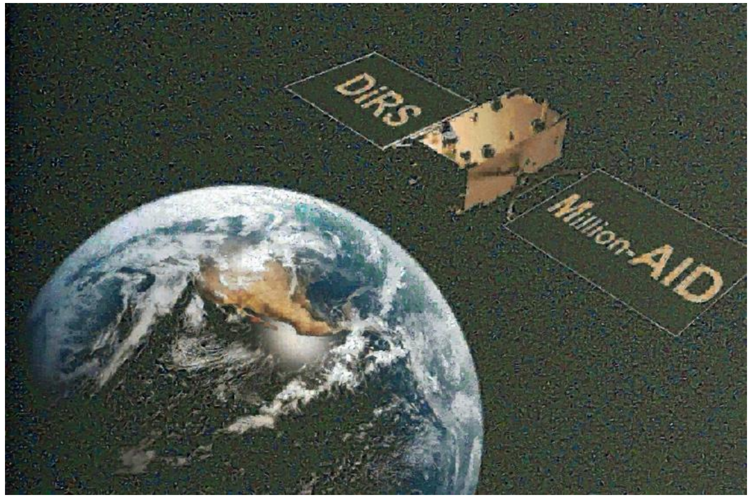
We are going to test six model sizes of a plain ViT

The first four models can fit on a single GCD (half of AMD-MI-250X) on Frontier

The ViT-5B can fit on two GCD’s and the ViT-15B can fit on four GCD’s

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720

Vit-MAE architecture on the Million-AID dataset



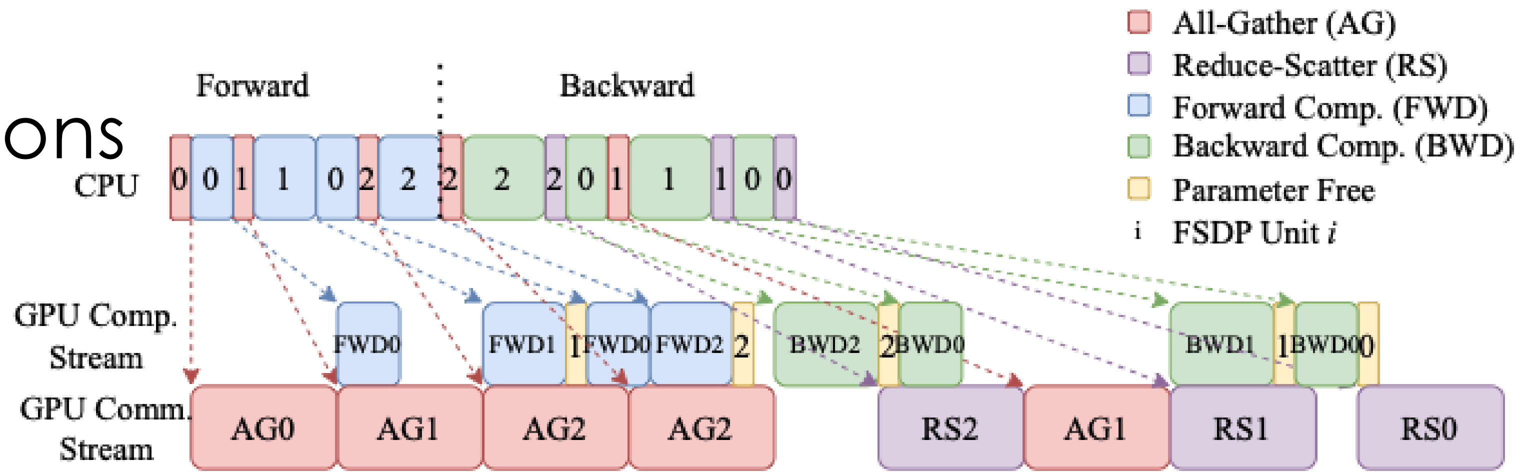
https://captain-whu.github.io/DiRS/

https://arxiv.org/pdf/2404.11706

FSDP Communication Option

Test Communication Options

- None
- BACKWARD PRE
- BACKWARD POST
- Limit all gathers




```
model = FSDP(model,
             mixed_precision=bfloatPolicy,
             sharding_strategy=sharding_strategy,
             device_id=torch.cuda.current_device(),
             use_orig_params=True,
             process_group=process_group,
             param_init_fn=my_init_fn,
             backward_prefetch=BackwardPrefetch.BACKWARD_PRE,
             #forward_prefetch=True,
             limit_all_gathers=True,
             )
```

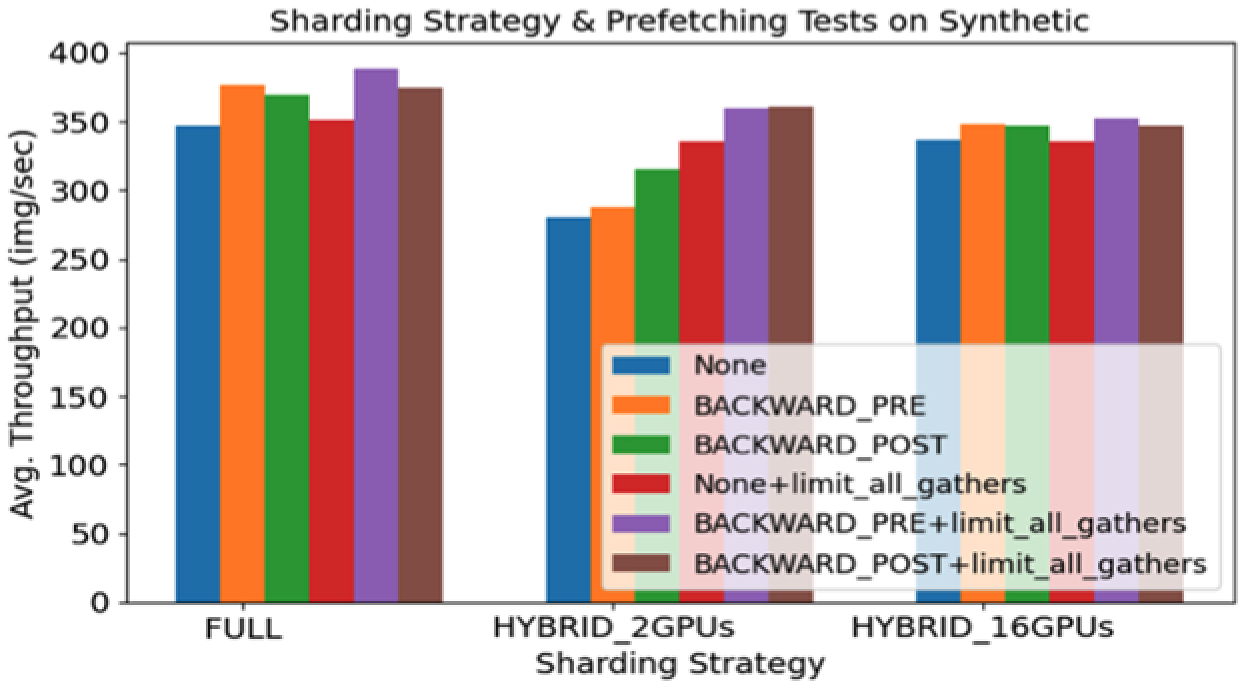
Fig. credit: https://arxiv.org/abs/2304.11277v2

FSDP Communication Option

Test Communication Options

- None
- BACKWARD PRE
- BACKWARD POST
- Limit all gathers

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720



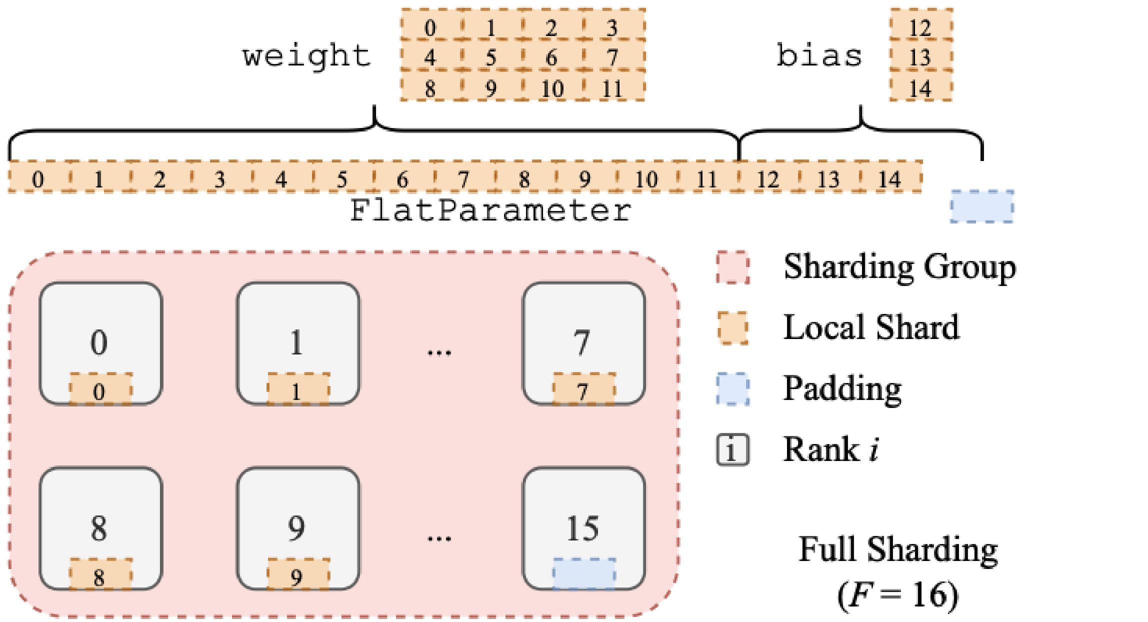
Best found BACKWARD PRE and limit-all-gathers (most compute-communication overlap)

https://arxiv.org/pdf/2404.11706

FSDP Model Sharding Options

Model Sharding Options

- FULL SHARD
- SHARD GRAD OP
- NO SHARD
- HYBRID SHARD



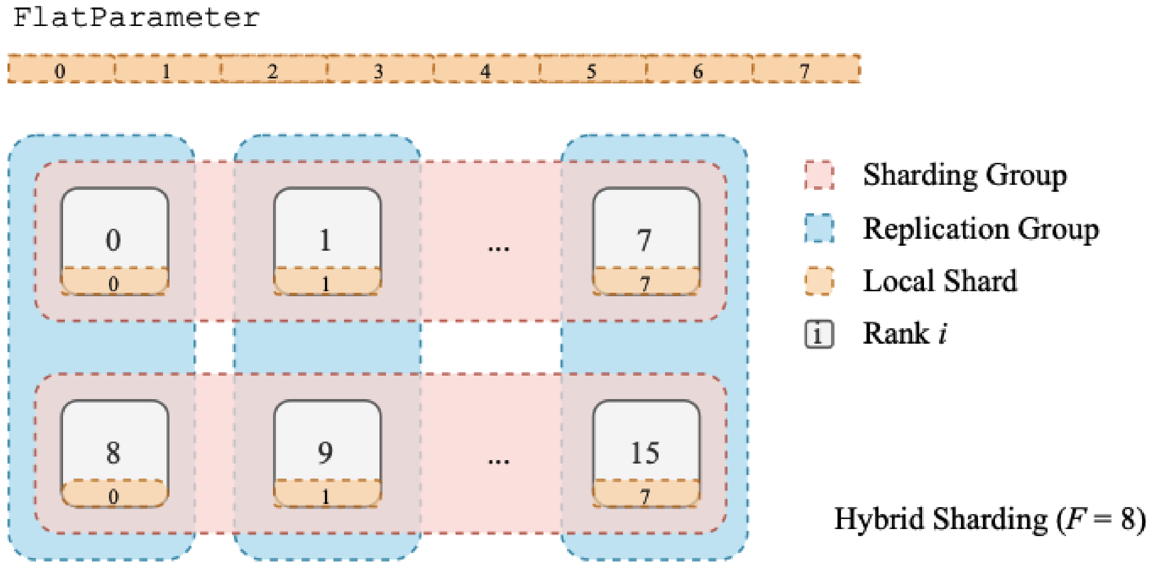


Fig. credit: https://arxiv.org/abs/2304.11277v2

FSDP Model Sharding Options

Model Sharding Options

- FULL SHARD
- HYBRID SHARD
- SHARD GRAD OP
- NO SHARD

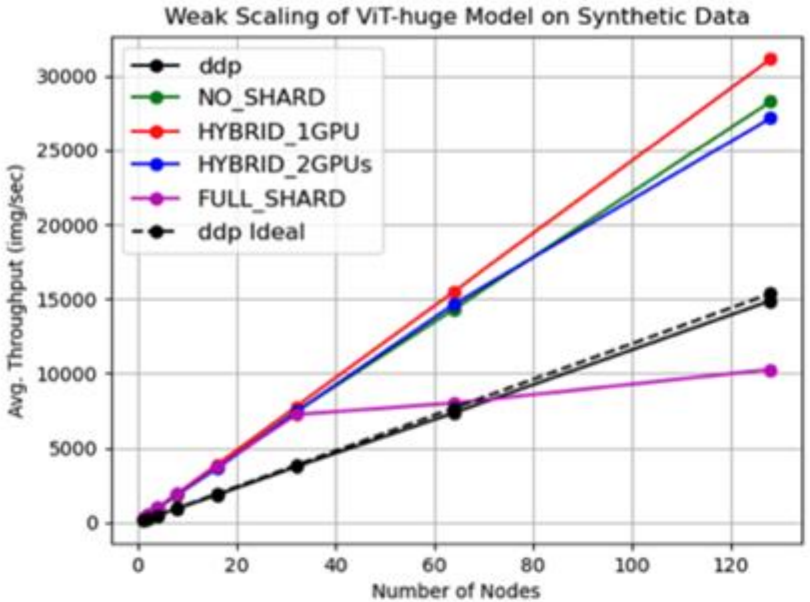
```python
sharding_strategy = ShardingStrategy.NO_SHARD
process_group = None
num_tasks = world_size
if (args.mode=='hybrid'):
    from torch.distributed._tensor import DeviceMesh

    scaling_group_size = 2  # how many gpus
    mesh_list = []
    dmesh = torch.arange(0, world_size).view(-1, scaling_group_size)
    for i, sg in enumerate(dmesh):
        mesh_list.append(sg.tolist())

    mesh = DeviceMesh(device_type="cuda", mesh=mesh_list)
    mesh_groups = mesh.get_dim_groups()
    replicate_group, shard_group = mesh_groups[0], mesh_groups[1]

    sharding_strategy = ShardingStrategy.HYBRID_SHARD
    process_group = (shard_group, replicate_group)
    num_tasks = world_size / scaling_group_size

model = FSDP(model,
             mixed_precision=bfloatPolicy,
             sharding_strategy=sharding_strategy,
             device_id=torch.cuda.current_device(),
             use_orig_params=True,
             process_group=process_group,
             param_init_fn=my_init_fn,
             backward_prefetch=BackwardPrefetch.BACKWARD_PRE,
             #forward_prefetch=True,
             limit_all_gathers=True,
             )
```

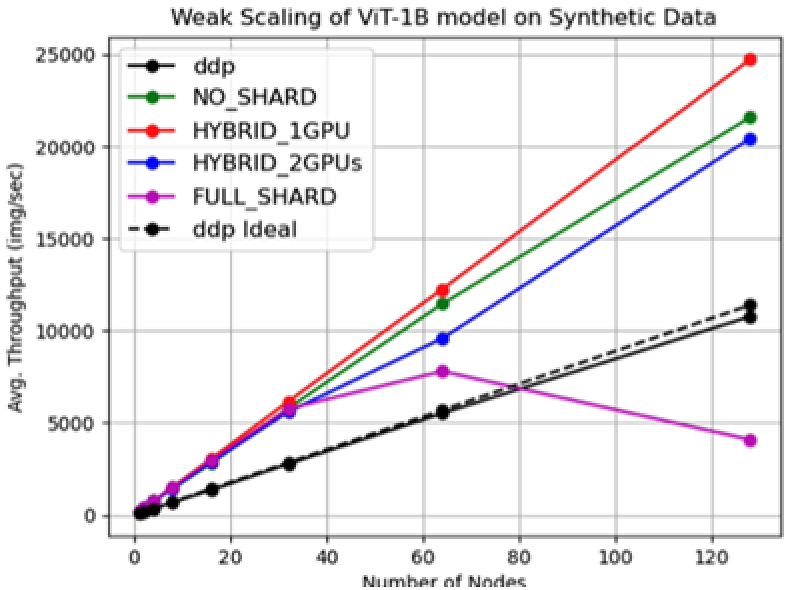
Fig. credit: https://arxiv.org/abs/2304.11277v2

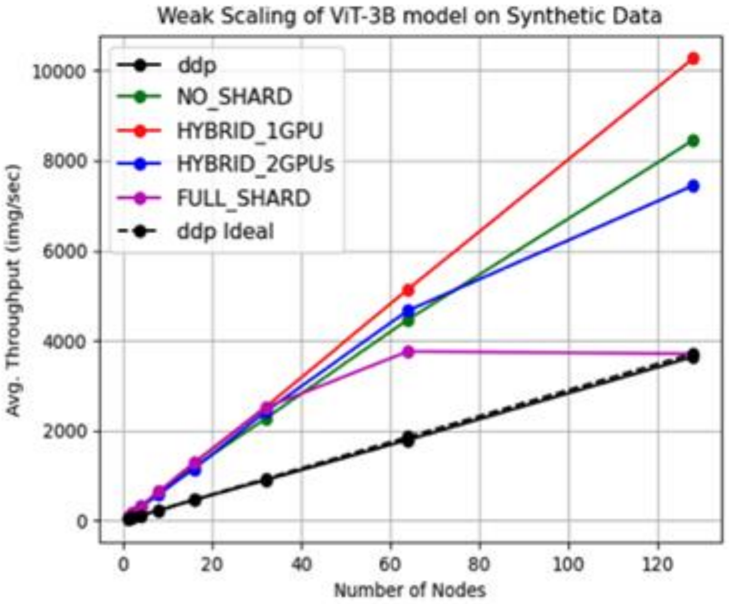
FSDP Model Sharding Options

Test scaling for models that can fit on single GCD: <u>throughput</u>

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720







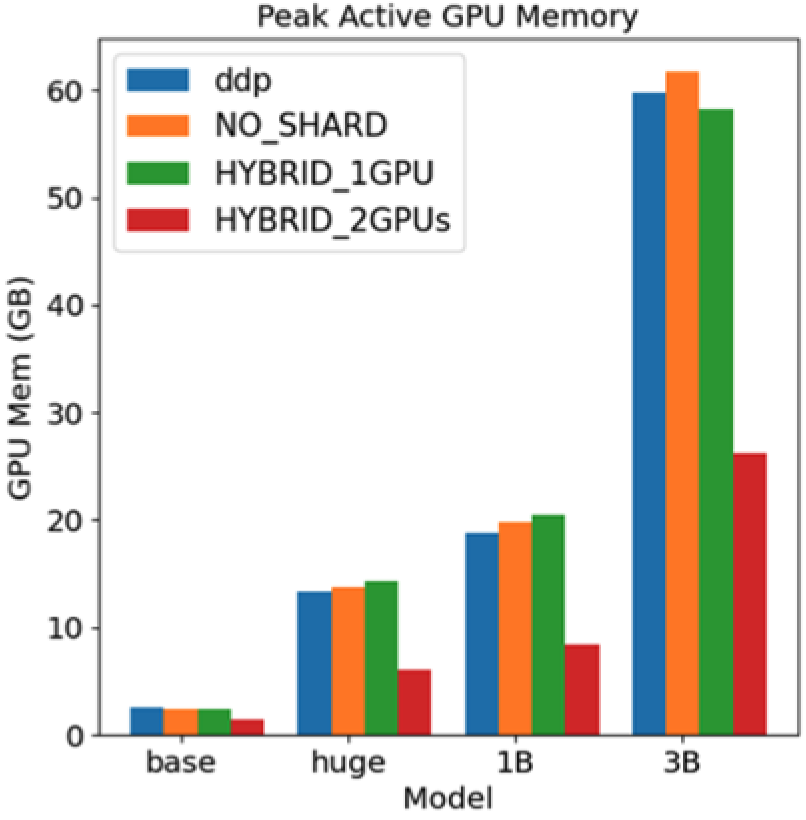
OAK RIDGE National Laboratory

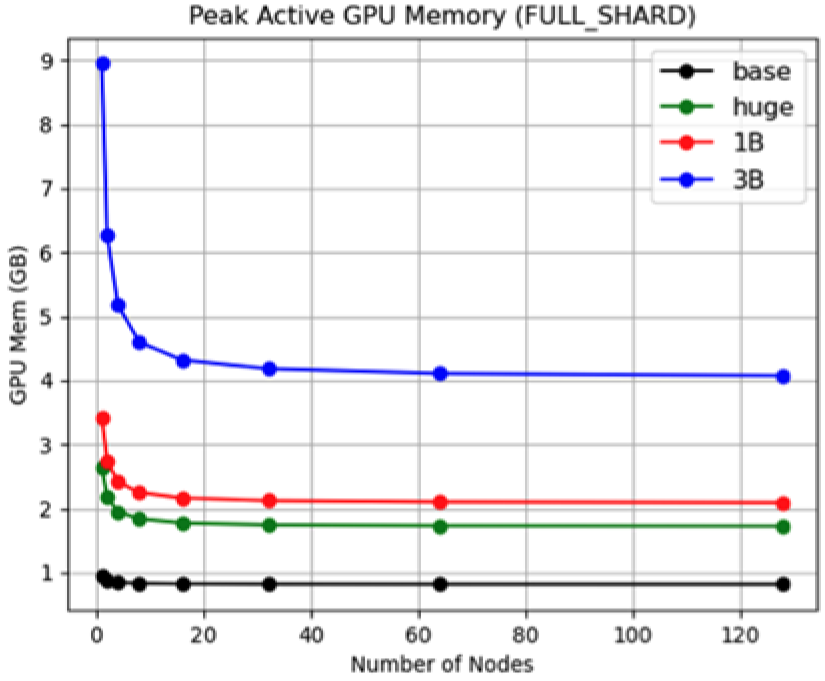
https://arxiv.org/pdf/2404.11706

FSDP Model Sharding Options

Test scaling for models that can fit on single GCD: <u>memory usage</u>

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720



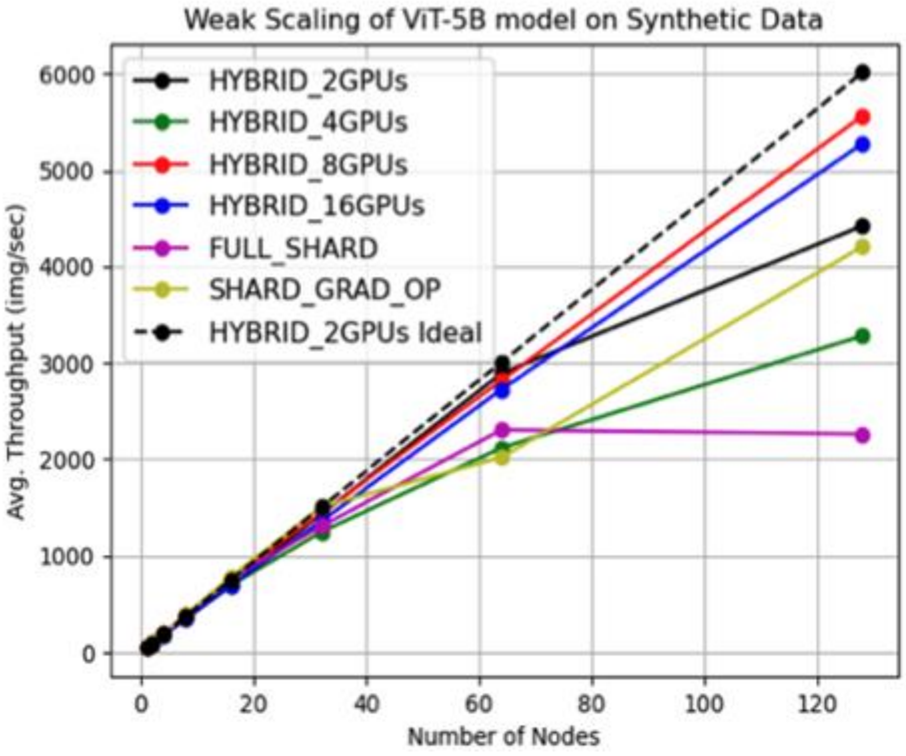


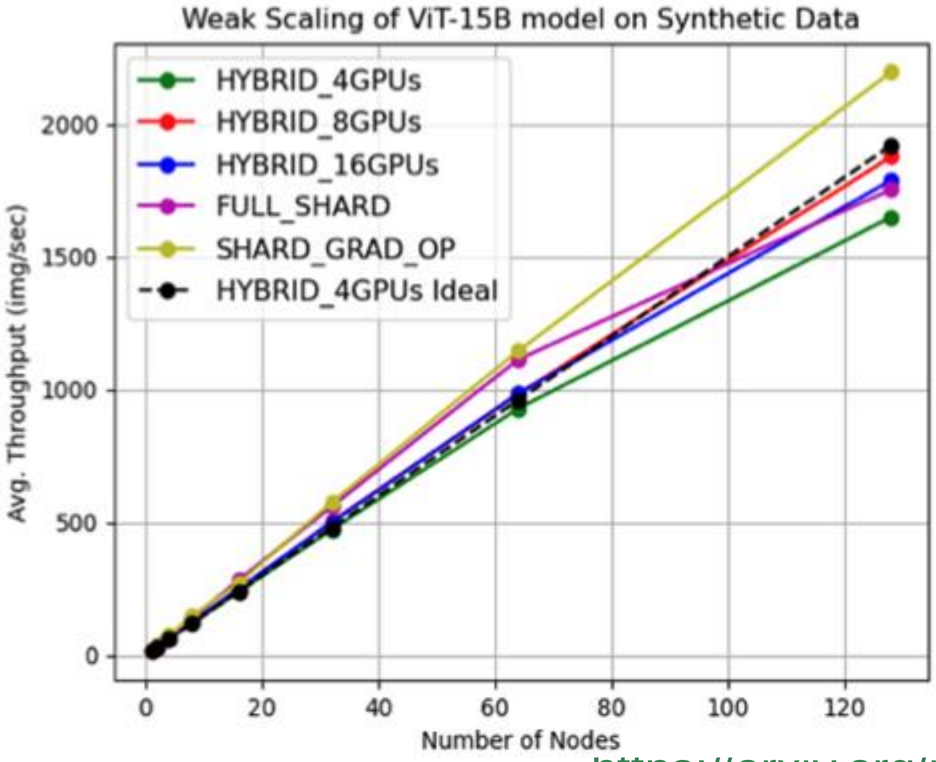
https://arxiv.org/pdf/2404.11706

FSDP Model Sharding Options

Test scaling for models that can't fit on single GCD: <u>throughput</u>

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720



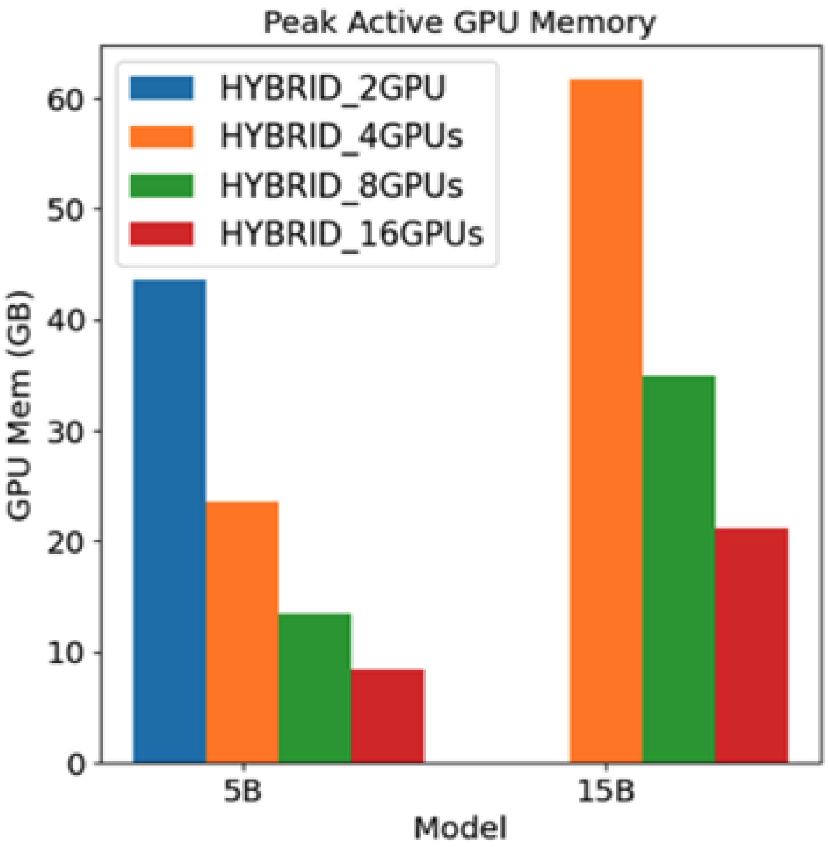


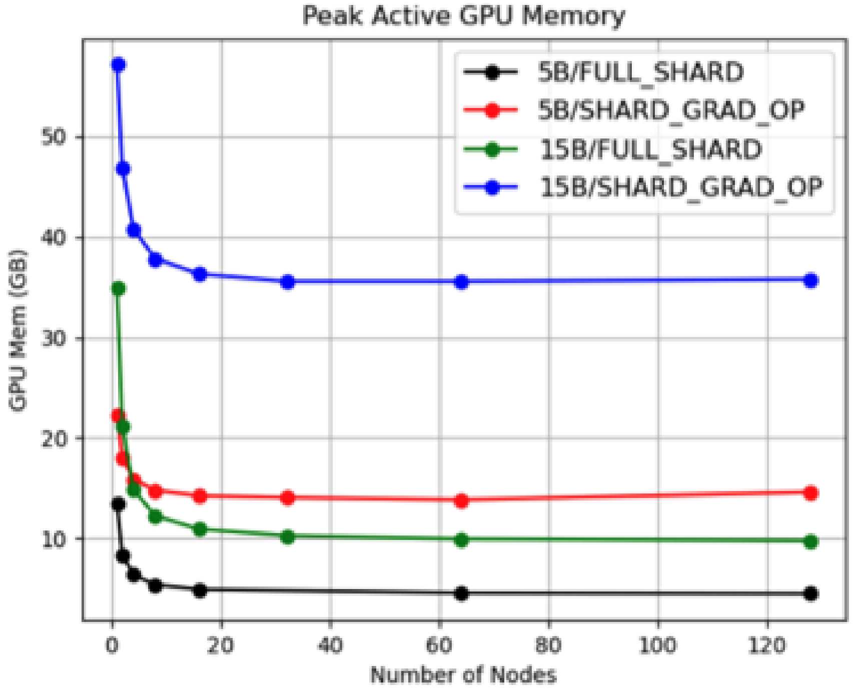
https://arxiv.org/pdf/2404.11706

FSDP Model Sharding Options

Test scaling for models that can't fit on single GCD: <u>memory usage</u>

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720





https://arxiv.org/pdf/2404.11706

Best Practice of I/O

It is a good practice to try to identify bottlenecks when scale. At first approach it is useful to just look real data Vs cached data Vs standalone data-loader run

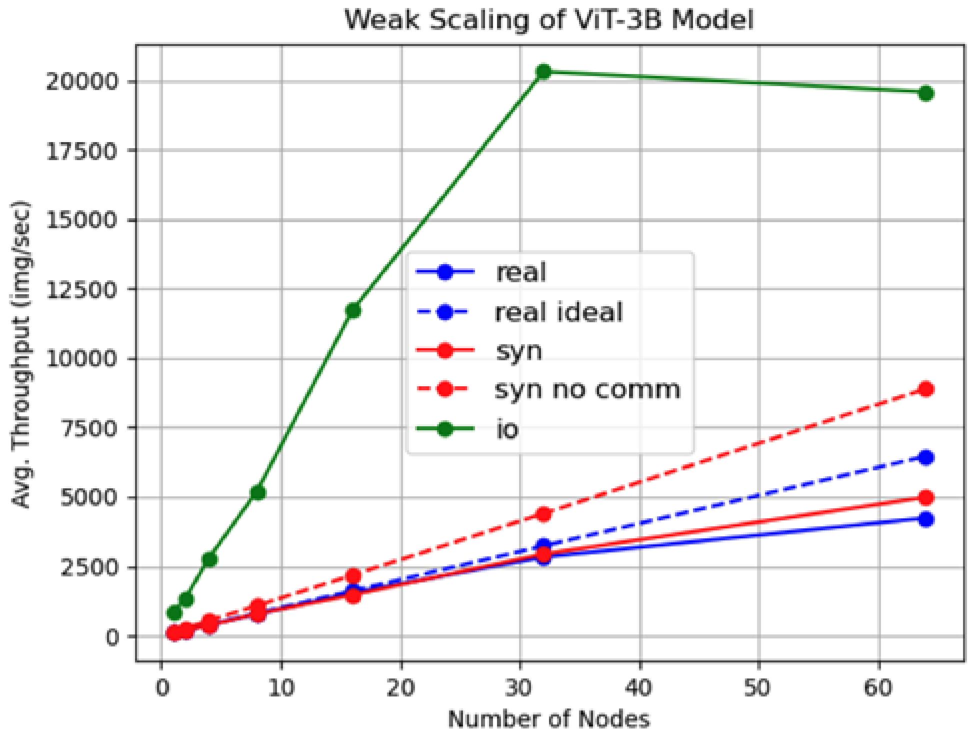
```
dataiter_train = iter(data_loader)
if(args.run == 'syn'):
    samples, _ = next(dataiter_train)
    samples = samples.to(torch.cuda.current_device())

for data_iter_step in range(0, len(data_loader)):

    if(args.run == 'real' or args.run == 'io'):
        samples, _ = next(dataiter_train)
        samples = samples.to(torch.cuda.current_device())

    if(args.run == 'real' or args.run == 'syn'):

        loss, _, _ = model(samples, mask_ratio=args.mask_ratio)

        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

Best Practice of I/O

Test if IO or communication is the main bottleneck, using remote-sensing images

Model	Width	Depth	MLP	Heads	Parameters [M]
ViT-Base	768	12	3072	12	87
ViT-Huge	1280	32	5120	16	635
ViT-1B	1536	32	6144	16	914
ViT-3B	2816	32	11264	32	3067
ViT-5B	1792	56	15360	16	5349
ViT-15B	5040	48	20160	48	14720

For the ViT-3B model it looks communication drives the performance rather than IO

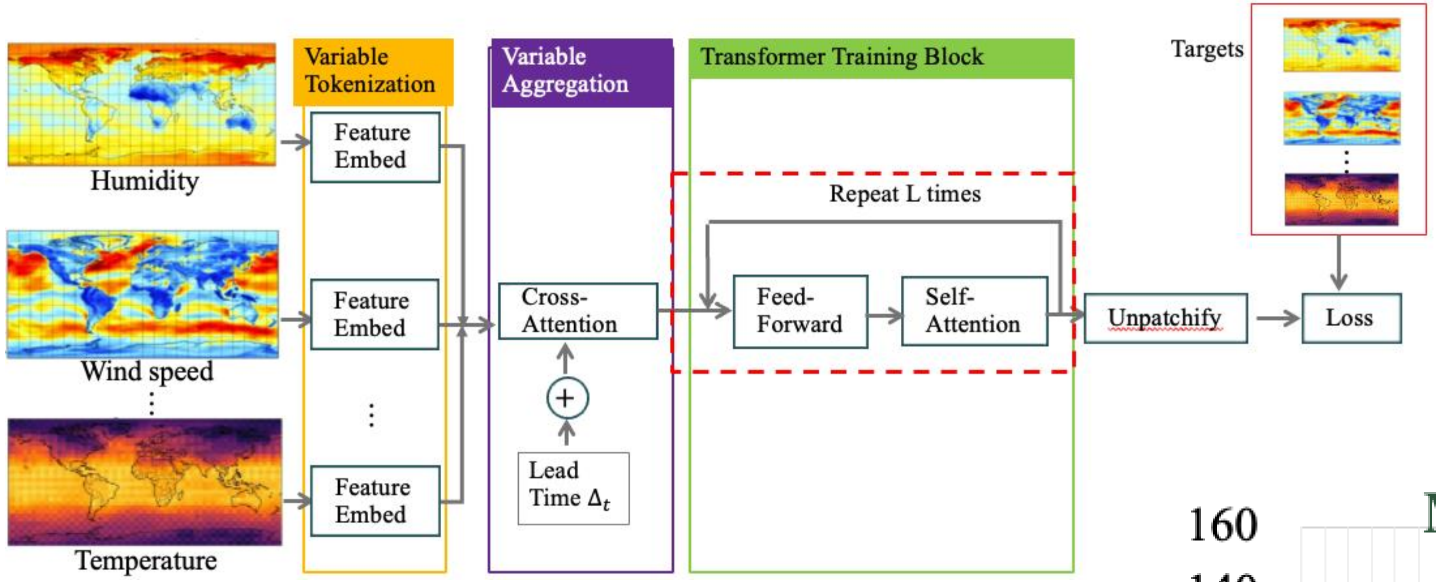


Also, you can estimate communication cost

Once bottleneck identified it needs detailed profiled tools to improve

https://arxiv.org/pdf/2404.11706

Example from Climate Forecasting: ORBIT



4D+X Climate Activity
longitude, latitude, time, (#height, #weather/climate variables) (91)

This is a ViT based architecture that combines FSDP with model parallelism (tensor-parallelism)

Limit of FSDP alone for this architecture

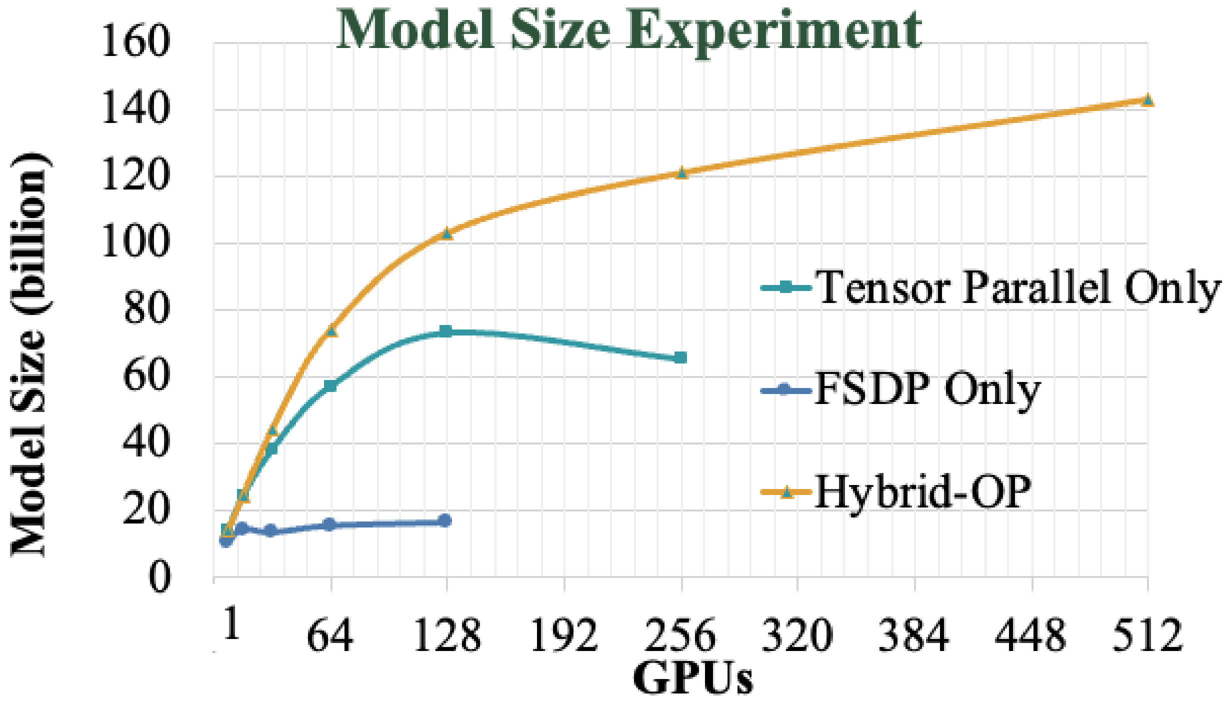


Fig. credit https://arxiv.org/abs/2404.14712v5

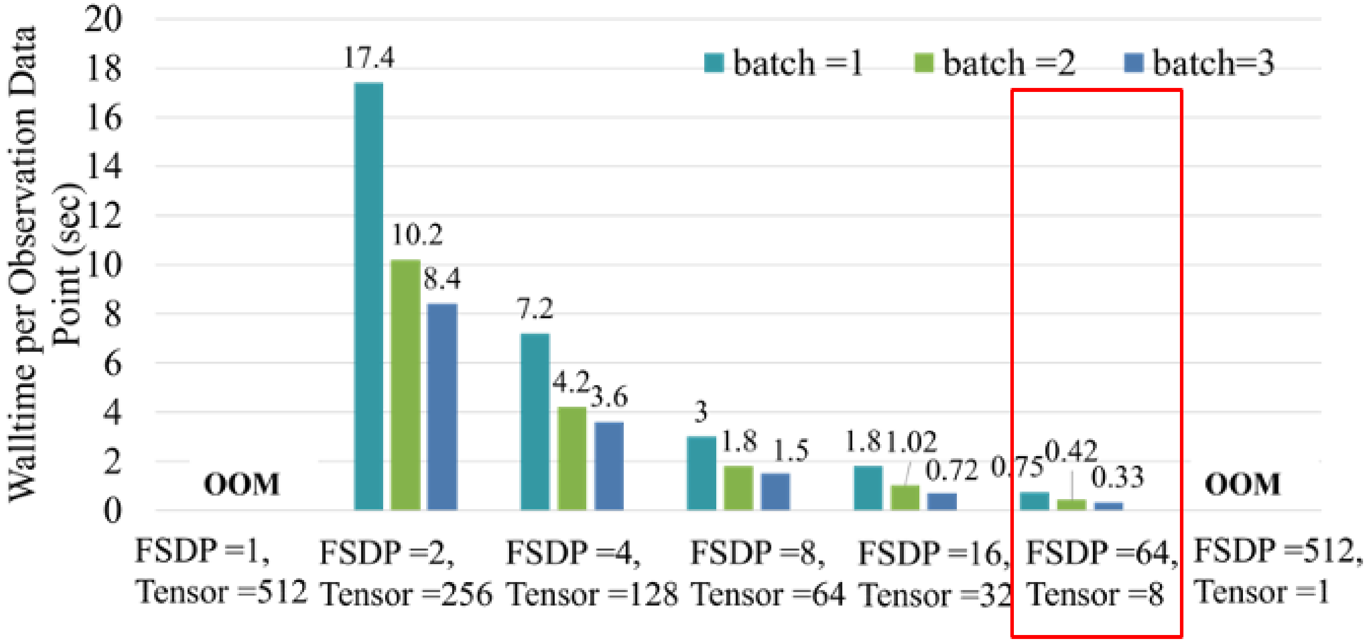
Oak Ridge National Laboratory

Example from Climate Forecasting: ORBIT

For 113 billion model using 512 GPUs we measure 5.7x speedup from these techniques

Layer wrapping	✗	✓	✓	✓	✓
Mixed precision	✗	✗	✓	✓	✓
Prefetching	✗	✗	✗	✓	✓
Activation Checkpoint	✗	✗	✗	✗	✓
Speedup	**OOM**	**1**	**1.97**	**2.4**	**5.7**

FSDP tuning and other optimizations help with more parallel modes as well



Usually, best performance with heavier communication within node and data-parallel across nodes

Fig. credit https://arxiv.org/abs/2404.14712v5

Thank you!