TUTORIAL

LLMOPS & MLOPS



Shagun Sodhani Tech Lead

Meta

Scaling Deep Learning Training with Fully Sharded Data Parallelism in PyTorch

ODSC[®] WEST

OCT 29-31 SAN FRANCISCO



Scaling Deep Learning Training with FSDP in PyTorch

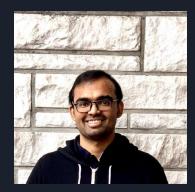
ODSC 2024

@shagunsodhani



About Me

- 1. Tech Lead and Staff Research Engineer @ Meta Al
- 2. Focused on building AI agents that can:
 - a. interact with and learn from the physical world
 - b. consistently improve as they do so without forgetting the previous knowledge





Agenda

- 1. What is scaling and why care about it
 - a. Challenges in scaling deep learning models
- 2. Fully Sharded Data Parallelism (FSDP)
 - a. Overview
 - b. FSDP in action



What is scaling

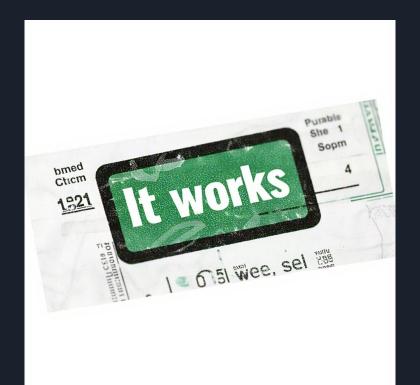
Scaling - Training larger models on larger datasets using more compute

Scaling Hypothesis - Scaling improves performance across diverse tasks.

Strong Scaling Hypothesis - Easiest way to optimize for all the tasks & data is to find a scalable architecture and simply train ever larger NNs [1]

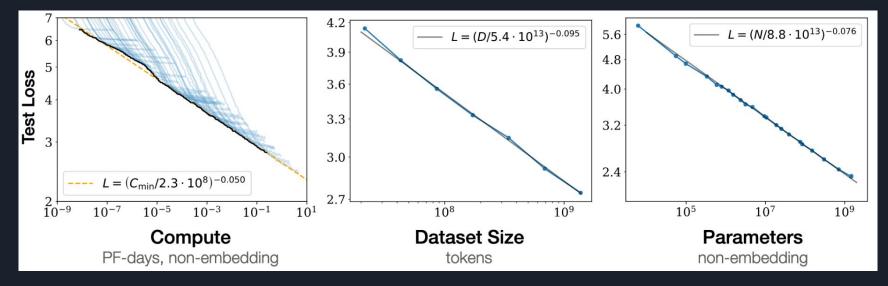


Why care about scaling





Why care about scaling



Taken from Scaling Laws for Neural Language Models [2]



Challenges in scaling deep learning models

- 1. Data Availability and Quality
- 2. Compute Resource Requirements
- 3. Cost of Training and Deployment
- 4. Long Experimentation Cycles
- 5. Energy Consumption and Environmental Impact
- 6. Talent and Expertise Gaps
- 7. ...



Challenges when training large models

- 1. Memory bottlenecks
 - a. Size of the model parameters + activations + optimizer state
- 2. Computation Efficiency
 - a. Parallelism overhead can reduce the expected speedups
- 3. Communication Overhead in Multi-Node Setups
 - a. Communication (e.g. for parameter update) limits scalability



Fully Sharded Data Parallelism









Data Parallelism



Model



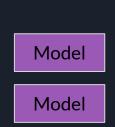
Data Parallelism





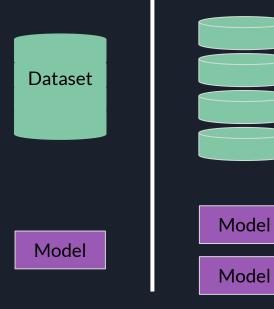
Data Parallelism







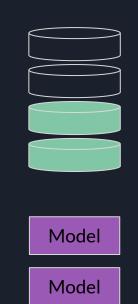
Data Parallelism



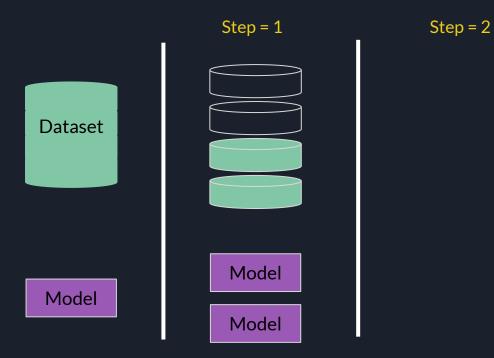


Data Parallelism

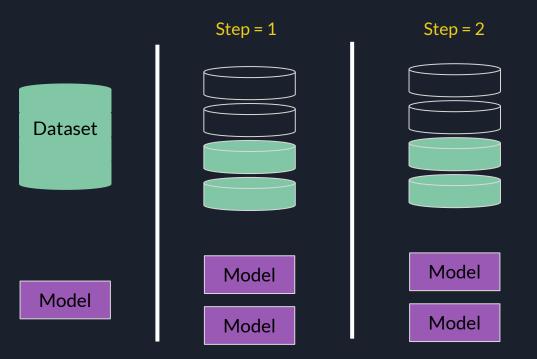




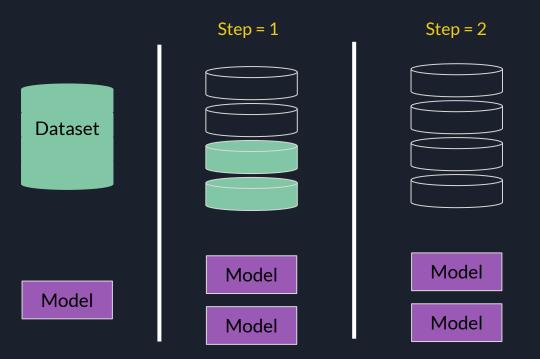














Data Parallelism

Each GPU has a full copy of the model

Split the dataset in batches and each gpu processes a different batch

Easy to use via **DistributedDataParallel**

Bottlenecked on the size of the model (or activations or optimizer state)

Inefficient for large models or lot of gpus



Fully Sharded Data Parallelism



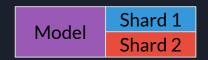


Sharded

Model



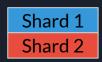
Sharded





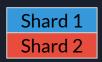
Sharded







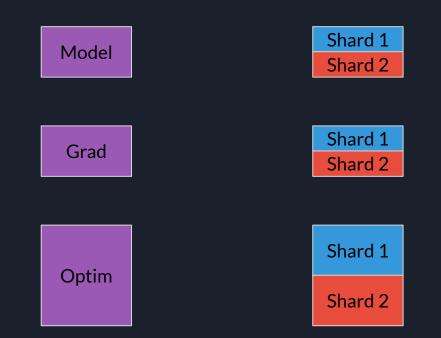














Fully Sharded Data Parallelism



Model



Fully Sharded Data Parallelism

Dataset Model



Fully Sharded Data Parallelism









Fully Sharded Data Parallelism

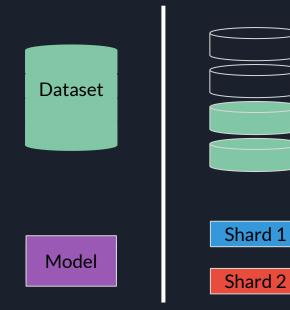
Dataset	
	Shard
Model	Shard
	- Offici C

Step = 1

2

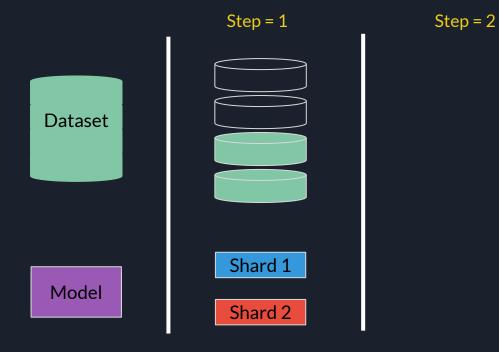


Fully Sharded Data Parallelism



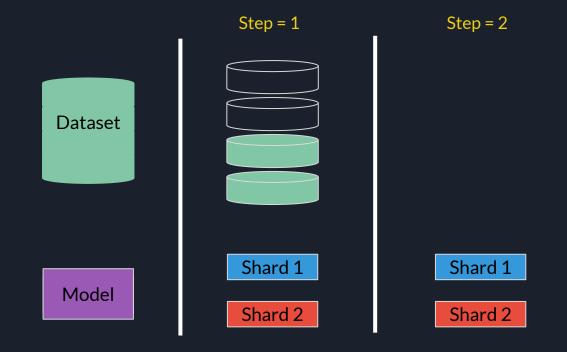


Fully Sharded Data Parallelism



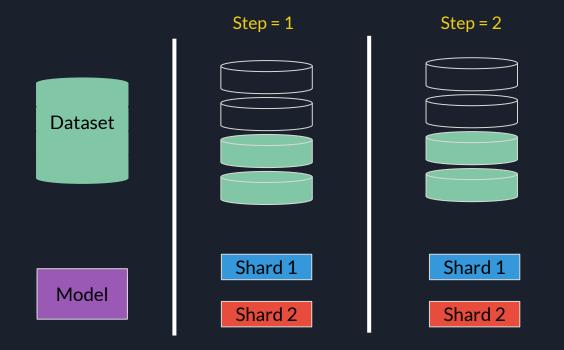


Fully Sharded Data Parallelism



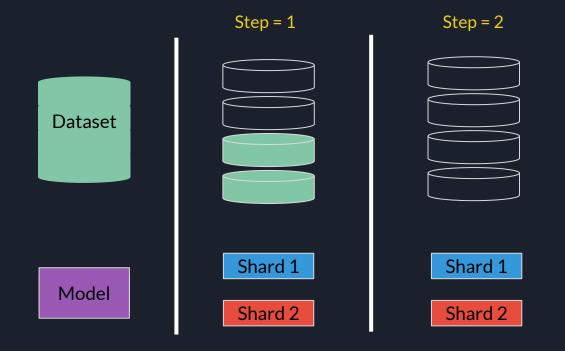


Fully Sharded Data Parallelism

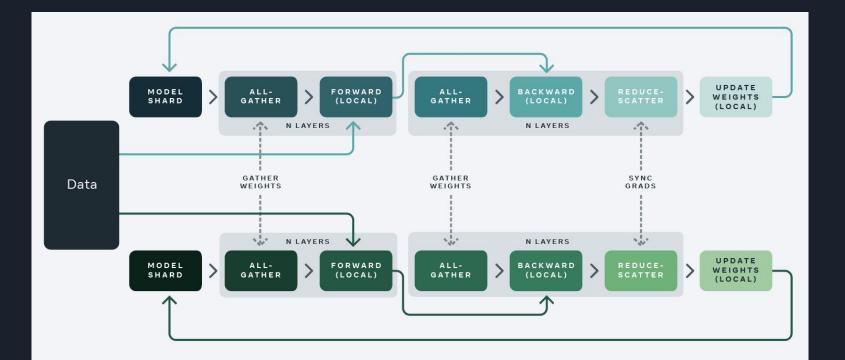




Fully Sharded Data Parallelism



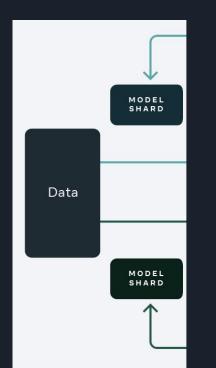




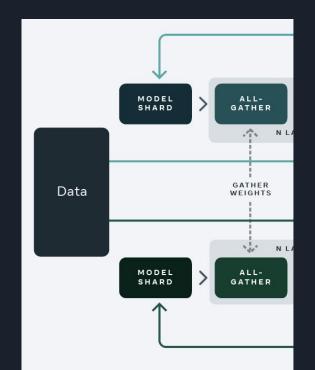


Data

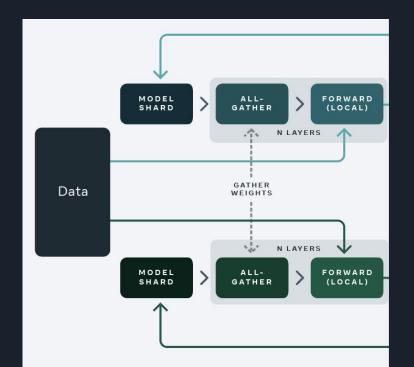




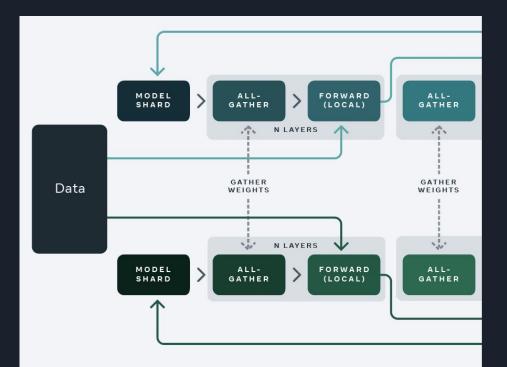




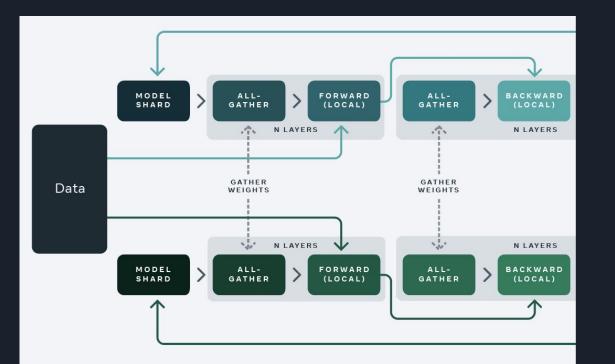




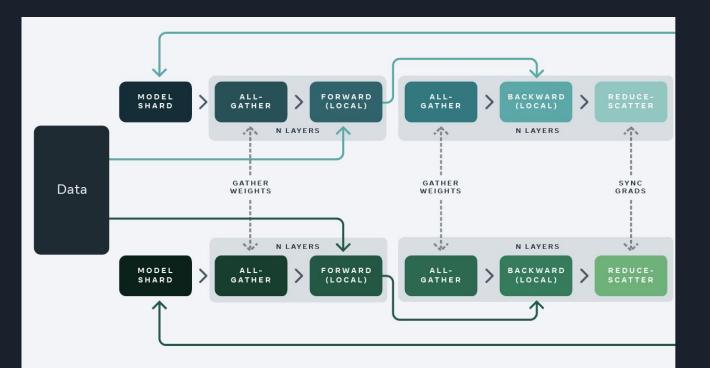




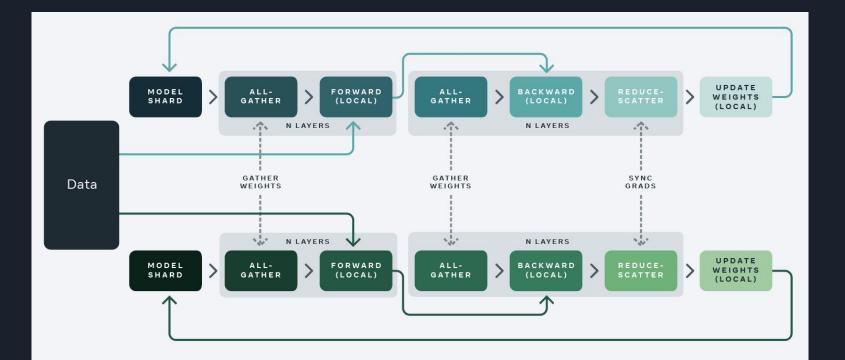














Fully Sharded Data Parallelism

Each GPU has a shard of the model (gradient and the optimizer state).

Split the dataset in batches and each gpu processes a different batch

Easy to use via *FullyShardedDataParallel* but less intuitive than *DistributedDataParallel*

Memory efficient but introduces more communication



model=MLP(

- (fcs): Sequential(
 - (0): Linear(in_features=20480, out_features=20480, bias=True)
 - (1): ReLU()
 - (2): Linear(in_features=20480, out_features=20480, bias=True)
 - (3): ReLU()
 - (4): Linear(in_features=20480, out_features=20480, bias=True)
 - (5): ReLU()
 - (6): Linear(in_features=20480, out_features=20480, bias=True)
 - (7): ReLU()
 - (8): Linear(in_features=20480, out_features=20480, bias=True)
 (9): ReLU()
 - (10): Linear(in_features=20480, out_features=20480, bias=True)
 (11): ReLU()
 - (12): Linear(in_features=20480, out_features=20480, bias=True)
 (13): ReLU()
 - (14): Linear(in_features=20480, out_features=20480, bias=True)



from torch.distributed.fsdp import FullyShardedDataParallel as FSDP

```
init_process_group(backend="nccl", rank=rank, world_size=world_size)
```

```
torch.cuda.set_device(rank)
device = torch.device(f"cuda:{rank}")
model = MLP()
model = model.to(device)
model = FSDP(model)
```



):

def train_step(

data: torch.Tensor, target: torch.Tensor, model: nn.Module, optimizer: optim.Optimizer, criterion: nn.Module,

```
optimizer.zero_grad()
output = model(data)
loss = criterion(output, target)
loss.backward()
optimizer.step()
```



- 1. Constructor
 - a. Shard model parameters and each rank only keeps its own shard



- 1. Constructor
 - a. Shard model parameters and each rank only keeps its own shard
- 2. Forward Call
 - a. *all_gather* all shards from all ranks to recover the full parameter in current FSDP unit
 - b. Run forward computation
 - c. Discard parameter shards it has just collected



- 1. Constructor
 - a. Shard model parameters and each rank only keeps its own shard

2. Forward Call

- a. *all_gather* all shards from all ranks to recover the full parameter in current FSDP unit
- b. Run forward computation
- c. Discard parameter shards it has just collected
- 3. Backward call
 - a. *all_gather* all shards from all ranks to recover the full parameter in current FSDP unit
 - b. Run backward computation
 - c. *Reduce_scatter* (sync) gradients
 - d. Discard parameters



FSDP in action | What to shard

FULL_SHARD - Parameters, gradients, and optimizer states are sharded.

NO_SHARD - Nothing is shared - This is very similar for DDP

HYBRID_SHARD - Apply FULL_SHARD within a node, and replicate parameters across nodes.



FSDP in action | What to shard

model = FSDP(
 model,
 sharding_strategy=ShardingStrategy.FULL_SHARD,



Specify a policy for sharding layers

The policy can be based on the size (number of parameters) of the model or name of the model

This is very easy to get wrong



model = FSDP(
 model,
 sharding_strategy=ShardingStrategy.FULL_SHARD,

Module View	
Module Name	
 FullyShardedDataParallel_0 	
- MLP_0	
 Sequential_0 	
Linear_0	
ReLU_0	
Linear_1	
ReLU_1	
Linear_2	
ReLU_2	
Linear_3	
ReLU_3	
Linear_4	
ReLU_4	
Linear_5	
ReLU_5	
Linear_6	
ReLU_6	
Linear_7	

Module View	
Module Name	
- FullyShardedDataParallel_0	
- MLP_0	
 Sequential_0 	
Linear_0	
ReLU_0	
Linear_1	nn Module: FullyShardedDataParailel_0
	torch/nn/modules/module.py(15555):_call_impl torch/distributed/sdp/tully_sharded_data_parallel.py(841): forward
ReLU_1	FullyShardedDataParallel.torward
Linear_2	torch/distributed/tsdp/_runtime_utils.py(342): _pre_forward
	FullyShardedDataParallel_pre_forward torch/distributed/fsdp/_runtime_utils.py(405):_pre_forward_unshard
ReLU_2	torcholatibuturdatapiutura_utura
Linear_3	FullyShardedDataParalleLate_limiter
Linear_3	torch/cuda/streams.py(216): synchronize
ReLU_3	dbull-in method synchronize of Event object at 0x710a73336760> cudaEventSynchronize
Linear_4	
ReLU_4	
Linear_5	
ReLU_5	
Linear_6	
ReLU_6	
Linear_7	

Module View	
Module Name	
 FullyShardedDataParallel_0 	
- MLP_0	
- Sequential_0	
Linear_0	
ReLU_0	
Linear_1	
ReLU_1	
Linear_2	
ReLU_2	
Linear_3	
ReLU_3	
Linear_4	
ReLU_4	
Linear_5	
ReLU_5	
Linear_6	
ReLU_6	
Linear_7	

Peak Memory Usage: 43218.2MB			
	nn.Module: FullyShardedDataPara	uliel_0	
	torch/nn/modules/module.py(1555): _	call_impl	
te	orch/distributed/fsdp/fully_sharded_data_paral		
	FullyShardedDataParallel.forwa		
	torch/distributed/fsdp/_runtime_utils.py(342		
	FullyShardedDataParallelpre_fo		
U	torch/distributed/fsdp/_runtime_utils.py(405): _pi		
	FullyShardedDataParallel.rate_lii		
	torch/cuda/streams.py(216): synch		
and the second	built-in method synchronize of Event object at		
	cudaEventSynchronize		



model = FSDP(
 model,
 sharding_strategy=ShardingStrategy.FULL_SHARD,
 auto_wrap_policy=ModuleWrapPolicy({nn.Linear}),

Module View
Module Name
- FullyShardedDataParallel_0
- MLP_0
 Sequential_0
FullyShardedDataParallel_1
Linear_0
ReLU_0
+ FullyShardedDataParallel_2
ReLU_1
+ FullyShardedDataParallel_3
ReLU_2
+ FullyShardedDataParallel_4
ReLU_3
+ FullyShardedDataParallel_5
ReLU_4
+ FullyShardedDataParallel_6
ReLU_5
+ FullyShardedDataParallel_7
ReLU_6
+ FullyShardedDataParallel_8

Module View

Module Name FullyShardedDat - MLP 0 - Sequential_ - FullyShar Linear

uk Name						
MLP_0 - Sequential_0 - Josepantial_0 - Sequential_0 - Sequential_0 - FullyShardedDataParallel_1 - Sequential_0 - ReLU_0 + FullyShardedDataParallel_2 - ReLU_1 + FullyShardedDataParallel_2 - ReLU_1 + FullyShardedDataParallel_3 - ReLU_3 - FullyShardedDataParallel_4 - FullyShardedDataParallel_5 - FullyShardedDataParallel_5 - FullyShardedDataParallel_5 - FullyShardedDataParallel_6 - Sult_4 - FullyShardedDataParallel_6 - Sult_4	ule Name					
- Sequential_0 - Sequential_0 - FullyShardedDataParallel_11 - Linear_0 ReLU_0 - FullyShardedDataParallel_2 - FullyShardedDataParallel_2 - FullyShardedDataParallel_2 - FullyShardedDataParallel_2 - FullyShardedDataParallel_3 - FullyShardedDataParallel_3 - FullyShardedDataParallel_4 - FullyShardedDataParallel_5 - FullyShardedDataParallel_5 - FullyShardedDataParallel_5 - FullyShardedDataParallel_6 - F	FullyShardedDataParallel_0					
FullyShardedDataParalleL1 torch/ns/modules/module.py(1555): _call_mpl Linear_0 torch/ns/modules/container.py(217): 10-ward ReLU_0 nn.Module: FullyShardedD_n_nModule: FullyShardedD	MLP_0					
Linear_0 Intercentinational explores include pyt (155);	- Sequential_0					
ReLU_0 nn.Module: FullyShardedD. nn.Module: FullyShardedD. nn.Module: FullyShardedD. FullyShardedDataParallel_2 torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nn/modules/modu torch/nimodules/modu torch/nimodules/modu torch/nimodules/modu torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ fullyShardedDataParal	FullyShardedDataParallel_1		torch/n	n/modules/module.py(1555): _(call_impl	
ReLU_0 torch/nn/modules/modu torch/ndistributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ fullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle FullyShardedDataParalle	Linear_0		torch/i	nn/modules/container.py(217):	forward	
i FullyShardedDataParallel_2 torch/nn/modules/modu torch/nn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/inn/modules/modu torch/instributed/fsdp/	Rel II 0	nn.Module: FullyS	nn.Module: FullyShardedD	nn.Module: FullyShardedD	nn.Module: FullyShardedD	1
ReLU_1 fullyShardedDataParalle_ FullyShardedDataParall_ FullySha		torch/nn/modules/modu	torch/nn/modules/modu	torch/nn/modules/modu	torch/nn/modules/modu	
+ FullyShardedDataParallel_3 torch/distributed/fsd torch/distributed/fsd torch/distributed/fsd ReLU_2 FullyShardedDataParallel	+ FullyShardedDataParallel_2	torch/distributed/fsdp/f	torch/distributed/fsdp/f	torch/distributed/fsdp/f	torch/distributed/fsdp/f	
+ FullyShardedDataParallel_3 FullyShardedDataParallel	ReLU_1	FullyShardedDataParal	FullyShardedDataParal	FullyShardedDataParal	FullyShardedDataParal	
ReLU_2 torch/distributed/fsdp/ torch/distribu	+ FullyShardedDataParallel_3	torch/distributed/fsd	torch/distributed/fsd	torch/distributed/fsd	torch/distributed/fsd	
i fullyShardedDataParalle_4 fullyShardedDataParalle_ <	Delli 2	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	
ReLU_3 torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ + FullyShardedDataParallel_5 torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ ReLU_4 torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ torch/distributed/fsdp/ + FullyShardedDataParallel_6 FullyShardedDataParallel FullyShardedDataParallel FullyShardedDataParallel FullyShardedDataParallel ReLU_5 torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams torch/cuda/streams ReLU_5 cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize	ReLU_Z	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	
Include	+ FullyShardedDataParallel_4	FullyShardedDataParall	FullyShardedDataParall	FullyShardedDataParall	FullyShardedDataParall	
+ FullyShardedDataParallel_5 ReLU_4 FullyShardedDataParallel_ FullyShardedDataParallel_ FullyShardedDataParallel_ FullyShardedDataParallel_ + FullyShardedDataParallel_6 forch/cuda/streams.py(216) forch/cuda/streams.py(216) forch/cuda/streams.py(216) ReLU_5 ReLU_5 cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize	ReLU_3	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	
ReLU_4 FullyShardedDataParallel FullyShardedDat		torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	
+ FullyShardedDataParallel_6 <th></th> <th>FullyShardedDataParallel</th> <th>FullyShardedDataParallel</th> <th>FullyShardedDataParallel</th> <th>FullyShardedDataParallel</th> <th></th>		FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	
ReLU_5 cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize cudaEventSynchronize	ReLU_4	to/ch/cuda/streams	torch/cuda/streams.py(216)	torch/cuda/streams.py(216)	torch/cuda/streams.py(216)	. 1
RéLU_S	+ FullyShardedDataParallel_6	 built-in method synchr	<tuilt-in method="" synchr<="" th=""><th><tuilt-in method="" synchr<="" th=""><th><built-in method="" synchr<="" th=""><th></th></built-in></th></tuilt-in></th></tuilt-in>	<tuilt-in method="" synchr<="" th=""><th><built-in method="" synchr<="" th=""><th></th></built-in></th></tuilt-in>	<built-in method="" synchr<="" th=""><th></th></built-in>	
+ FullyShardedDataParallel_7	ReLU_5	cudaEventSynchronize	cudaEventSynchronize	cudaEventSynchronize	cudaEventSynchronize	
	+ FullyShardedDataParallel_7					

nn.Module: FullyShardedD..

torch/nn/modules/modu...

torch/distributed/fsdp/f...

FullyShardedDataParal...

torch/distributed/fsd...

FullyShardedDataParallel...

torch/distributed/fsdp/_...

FullyShardedDataParall...

torch/distributed/fsdp/_...

torch/distributed/fsdp/_run...

FullyShardedDataParallel...

torch/cuda/streams.py(216) ...

<built-in method synchr...

cudaEventSynchronize

- ReLU_6
- + FullyShardedDataParallel_8

Module View

Module Name - FullyShardedDataParallel_0 - MLP_0

- Sequential_0

FullyShardedDataParallel_1

Linear_0

ReLU_0

+ FullyShardedDataParallel_2

ReLU_1

+ FullyShardedDataParallel_3

ReLU_2

+ FullyShardedDataParallel_4

ReLU_3

+ FullyShardedDataParallel_5

ReLU_4

+ FullyShardedDataParallel_6

ReLU_5

+ FullyShardedDataParallel_7

ReLU_6

+ FullyShardedDataParallel_8

Peak Memory Usage: 17616.6MB

torch/nn/modules/module.py(1555): _call_impl					
	torch/nn/modules/container.py(217): forward				
nn.Module: FullyS	nn.Module: FullyShardedD	nn.Module: FullyShardedD	nn.Module: FullyShardedD	nn.Module: FullyShardedD	
torch/nn/modules/modu	torch/nn/modules/modu	torch/nn/modules/modu	torch/nn/modules/modu	torch/nn/modules/modu	
torch/distributed/fsdp/f	torch/distributed/fsdp/f	torch/distributed/fsdp/f	torch/distributed/fsdp/f	torch/distributed/fsdp/f	
FullyShardedDataParal	FullyShardedDataParal	FullyShardedDataParal	FullyShardedDataParal	FullyShardedDataParal	
torch/distributed/fsd	torch/distributed/fsd	torch/distributed/fsd	torch/distributed/fsd	torch/distributed/fsd	
FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	
torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	
FullyShardedDataParall	FullyShardedDataParall	FullyShardedDataParall	FullyShardedDataParall	FullyShardedDataParall	
torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	torch/distributed/fsdp/	
torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	torch/distributed/fsdp/_run	
FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	FullyShardedDataParallel	
torch/cuda/streams	torch/cuda/streams.py(216)	torch/cuda/streams.py(216)	torch/cuda/streams.py(216)	torch/cuda/streams.py(216)	
<built-in method="" synchr<="" td=""><td><built-in method="" synchr<="" td=""><td><tuilt-in method="" synchr<="" td=""><td>cuilt-in method synchr</td><td><built-in method="" synchr<="" td=""></built-in></td></tuilt-in></td></built-in></td></built-in>	<built-in method="" synchr<="" td=""><td><tuilt-in method="" synchr<="" td=""><td>cuilt-in method synchr</td><td><built-in method="" synchr<="" td=""></built-in></td></tuilt-in></td></built-in>	<tuilt-in method="" synchr<="" td=""><td>cuilt-in method synchr</td><td><built-in method="" synchr<="" td=""></built-in></td></tuilt-in>	cuilt-in method synchr	<built-in method="" synchr<="" td=""></built-in>	
cudaEventSynchronize	cudaEventSynchronize	cudaEventSynchronize	cudaEventSynchronize	cudaEventSynchronize	



FSDP in action | CPU Offload

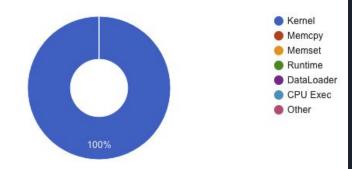
model = FSDP(
 model,
 sharding_strategy=ShardingStrategy.FULL_SHARD,
 auto_wrap_policy=ModuleWrapPolicy({nn.Linear}),
 cpu_offload=CPU0ffload(offload_params=True),



FSDP in action | CPU Offload = False

Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	52,133,082	100
Kernel	52,132,582	100
Memcpy	0	0
Memset	10	0
Runtime	0	0
DataLoader	0	0
CPU Exec	275	0
Other	216	0
4 9		



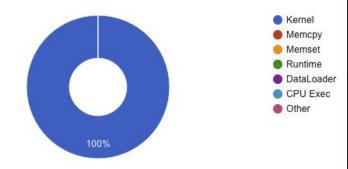


FSDP in action | CPU Offload = False

Peak Memory Usage: 17616.6MB

Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	52,133,082	100
Kernel	52,132,582	100
Memcpy	0	0
Memset	10	0
Runtime	0	0
DataLoader	0	0
CPU Exec	275	0
Other	216	0
4 9		





FSDP in action | CPU Offload = True

Peak Memory Usage: 16016.6MB

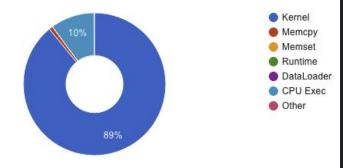


FSDP in action | CPU Offload = True

Peak Memory Usage: 16016.6MB

Execution Summary

Category	Time Duration (us)	Percentage (%)
Average Step Time	62,136,475	100
Kernel	55,323,892	89.04
Memcpy	531,020	0.85
Memset	11	0
Runtime	0	0
DataLoader	0	0
CPU Exec	6,197,215	9.97
Other	84,338	0.14
4 8		





FSDP in action | Other Options

- 1. forward_prefetch
- 2. limit_all_gathers / rate limiter
- 3. mixed_precision



FSDP in action | Profiling

- 1. Standard Pytorch profiling techniques apply [5, 6]
- 2. Look out for
 - a. time spent in sharding or unsharding parameters during forward and backward passes.
 - b. How often and how long all-gather operations take to complete, especially across multiple nodes.

FSDP in action | Common Pitfalls

- 1. Ensure consistent initialization using say *sync_module_states*
- 2. Use *backward_prefetch*, *forward_prefetch* and *limit_all_gathers* to reduce network latency
- 3. Use cpu_offload, mixed_precision and activation checkpointing to reduce memory usage
- 4. Uneven GPU Loads due to uneven sharding of model
- 5. <u>Checkpointing the models</u>



Next Step

- 1. <u>Getting Started with FSDP PyTorch Tutorials</u>
- 2. <u>Advanced Model Training with FSDP PyTorch Tutorials</u>
- 3. <u>PyTorch FSDP Tutorials YouTube</u>



Beyond FSDP

- 1. <u>FSDP2</u>
- 2. <u>Pipeline Parallel</u>
- 3. <u>Context Parallel</u>
- 4. <u>Tensor Parallel</u>



References

- 1. <u>Strong Scaling Hypothesis</u>
- 2. <u>Scaling Laws for Neural Language Models</u>
- 3. <u>DeepSpeed</u>
- 4. FSDP vs DeepSpeed
- 5. <u>PyTorch Profiler PyTorch Tutorials</u>
- 6. torch.profiler PyTorch 2.5 documentation

Thank you!

https://shagunsodhani.com/talks/

@shagunsodhani

ODSC 2024