Hydra: Efficient Training for Larger-Than-Memory Deep Learning Models
Kabir Nagrecha | Advised by Arun Kumar

**Problem**

GPU memory is limited...
Deep Learning models are growing rapidly!

**What's Needed:** An efficient platform for distributed training of large models

**Bottleneck:** Traditional Model Parallelism uses multiple devices to handle the memory demands of a single model. But this reduces our ability to parallelize compute!

---

**Hydra: Model Spilling, Shard Alternator Parallelism, and Double Buffering**

**Model Spilling**
Detach training orchestration from GPU arrangement

**Shard Alternator Parallelism (SHARP)**
Blend model and task parallelism for high throughput training

**Double Buffering**
Overlap communication with compute for low latency training

---

**Evaluation**

**Benchmark Dataset:**
WikiText-2

**Workload:**
Model Selection
12 1B+ parameter models
Transformer pretraining task
8-32 batch size
128 sequence length

**Hardware:**
Single-node, 8 12GB GPUs

**End-to-End Runtime Speedup (8 devices)**
- PyTorch Native: 1.0X
- Microsoft DeepSpeed: 1.3X
- FlexFlow (Model Parallel): 1.93X
- Hydra (SHARP): 7.46X

**Total GPU Utilization**
- PyTorch Native: 7%
- Microsoft DeepSpeed: 26%
- FlexFlow (Model/Data Parallel): 23%
- Hydra (SHARP): 82%

**Hydra produces near-optimal speedups!**
- 82% Average GPU Utilization
- >7.4X Speedups with 8 Devices

---

**Ongoing Work & Potential Impact**

Data Parallelism
User Study - Deep Learning for Physical Simulations
Concurrent Training at Scale