Piper: Multidimensional Planner for DNN Parallelization

Zillion-dollar question: how to train DNNs efficiently?

**Dimension 1:**
- **Data parallelism:**
  - Replicate model on every worker
  - Train on disjoint samples

  **But:**
  - Communication (weight resync) can be expensive
  - SOTA models are huge and don’t fit on a single worker

**Dimension 2:**
- **Model parallelism:**
  - Partition the model
  - Transfer intermediate activations between workers

To get high worker utilization, use **pipelining:**
- Once the first sample goes to Machine 2, Machine 1 can start processing the second sample, etc.

**Dimension 3:**
- **Tensor (model) parallelism (intra-layer):**
  - Can also split individual layers and operators for the same microbatch/sample
  - Think of matrix multiplication: many ways to split matrices
  - Scheme proposed by nVidia for Transformers (Megatron-LM)

  **Advantages:**
  - does not increase batch size
  - with only data parallelism and pipeline model parallelism, batch size × microbatch size × number of devices
  - can have smaller memory usage
  - indispensable if single layer doesn’t fit on 1 device

**Dimension 4:**
- **Memory-saving optimizations** such as activation recomputation

---

Prior work

**Approach 1:**
- Treat objective function as black box, e.g. measure time of 10 training steps
- Optimize it with generic heuristics such as Reinforcement Learning or MCMC
  - [Mirhoseini et al. 2017, 2018]
  - [Gao et al. 2018]
  - [Addanki et al. 2019]
  - [Zhou et al. 2019, 2020]
  - [Paliwal et al. 2020]

**Approach 2 (ours):**
- Build cost model that closely reflects real performance
- Solve resulting “offline” optimization problem with principled algorithmic techniques
  - [Jia et al. 2018, 2019]
  - [Narayanan et al. 2019, 2020]
  - [Tarnawski et al. 2020]

---

**Our two-level approach**

**Huge search space,** incl. finding good tensor parallelism schemes for entire DNN operator graph

- **Find good tensor parallelism schemes for individual layer types**
- **Combine them well, together with the other modes of parallelism**

---

**Our main contribution:** Piper, an efficient algorithm for this problem

---

**Our findings**

- We evaluate Piper on several modern DNN workloads
- Piper is efficient
- Piper beats out planners from prior work (PipeDream, PipeDream-2BW)
  - Tensor parallelism very useful with very large number of devices
  - Heterogeneous stages are advantageous, even for very repetitive DNNs

**Future work**

- Piper can handle PipeDream and PipeDream-2BW schedules; TODO: take pipeline flushes into account
- Most importantly, solve THIS problem

---

Some figures courtesy of PipeDream / Megatron

---

Jakub Tarnawski, Deepak Narayanan, Amar Phanishayee
Microsoft Research