Summary of Efficient Large-Scale Pre-Training - Pathways and PTD-P in Megatron-LM

Daniel Gorelik (djgoreli), Norman Wen (normanqw), Zhixiang Teoh (zhteoh)

Problem and Motivation

One of the common problems we have seen in generative AI systems so far is designing a system that can efficiently account for the sheer size of LLMs. In this lecture, we approached this question through the lens of pre-training—specifically, how do you efficiently pre-train a large-scale language model? This is a challenging problem for several reasons, the most significant being that GPU memory capacity is not large enough to store a full-scale LLM and that the necessary number of compute operations result in unrealistically long training times.

The systems that we analyze address this problem from slightly different angles on different hardwares. The motivation behind *Megatron-LM* is how to combine different parallelism techniques to optimize the training throughput on NVIDIA GPUs. On the other hand, *Pathways* explores scheduling and resource allocation techniques to accelerate pre-training on TPUs. Both target systems-level optimizations that preserve all computations of the models.

Related Works

Multi-controller architectures. There are several examples of distributed ML systems similar to *Pathways* with multi-controller architectures. These include Message Passing Interface (MPI), PyTorch, later versions of TensorFlow, and JAX. The main upside to these architectures is the low latency for dispatching accelerator computations, however they are not as well-suited for pipelining or sparse workloads.

DeepSpeed ZeRO. DeepSpeed ZeRO Redundancy Optimizer is a group of memory optimization technologies for large-scale distributed deep learning. Similar to *Megatron-LM*, ZeRO offers sharded data parallelism that can be combined with model parallelism to potentially improve its scaling behavior. ZeRO extends the idea of data parallelism by also sharing weight parameters and gradients across data-parallel workers.

Solution Overview

Pathways

Pathways is a single-controller architecture that offers a flexible programming model to express multiple programs, multiple data (MPMD) computations, with comparable performance to multi-controller architectures.

The *Pathways* system consists of accelerator "islands" interconnected by datacenter network (DCN), with intra-island connectivity via high-bandwidth intra-cluster interconnect (ICI). A global

centralized resource manager manages devices across islands, and dynamically maps physical to virtual devices satisfying the user's desired network and computation requirements. This layer of indirection between physical and virtual decouples user programs from resource management and enables multi-tenant resource sharing, similar to the benefits of virtual memory in operating systems.

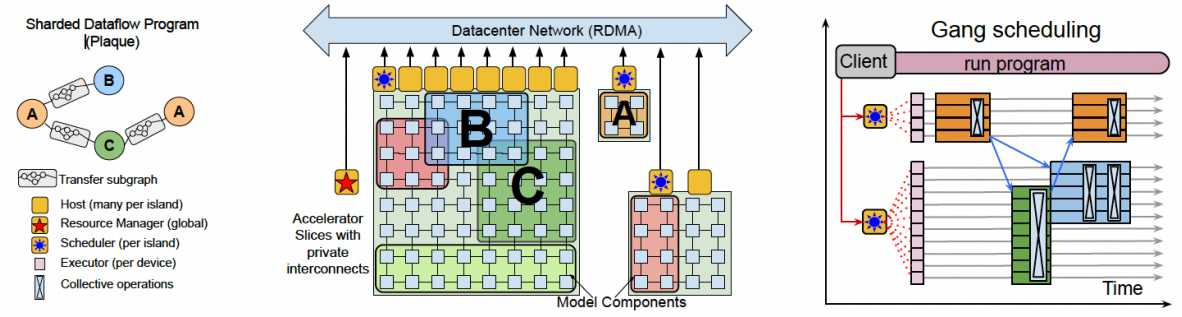

Figure 1. *Pathways* system overview

With *Pathways*, users simply wrap Python code with decorators to indicate fragments that should be compiled into single program, multiple data (SPMD) computations with known resource requirements, called "compiled functions", that are assigned virtual devices and each mapped to a single sharded computation node in a *Pathways* PLAQUE dataflow graph. To avoid the DCN becoming a bottleneck for data-dependent control flow on accelerators, *Pathways'* coordination runtime supports efficient sparse data exchanges along sharded transfer edges in the dataflow graph.

On the backend, *Pathways'* centralized scheduler per island schedules related tasks together. This gang-scheduling enhances throughput, minimizing context switches and preventing deadlocks due to TPU's lack of local preemption. *Pathways* employs parallel asynchronous dispatch to overlap computation with communication and pipeline tasks, by pre-allocating resources for computations with known resource requirements. This is especially applicable when computation time is much smaller than scheduling time, and to large training workloads. This approach suits TPUs well due to their limited support for dynamic matrix shapes.

Results. Evaluations against single-controller (TF, Ray) and multi-controller (JAX) systems show *Pathways* scales effectively, maintaining low dispatch and context-switching overheads and high throughput, even as pipeline stages increase, demonstrating its efficiency in managing large-scale training workloads on thousands of TPUs.

PTD-P in Megatron-LM

Pipeline, tensor, and data parallelism (PTD-P) is a technique that leverages the combination of pipeline parallelism across multi-GPU servers, tensor parallelism within a multi-GPU server, and data parallelism, to gracefully enable efficiently training GPT models with billions of parameters.

Pipeline model parallelism. PTD-P refined pipeline model parallelism to tackle common inefficiencies associated with traditional approaches. At the end of each minibatch, when switching between forward passes and backward passes, pipelines need to be flushed in order to retain optimizer semantics, resulting in “pipeline bubbles”. PTD-P takes advantage of sophisticated interleavings and model sharding to reduce these pipeline bubbles.

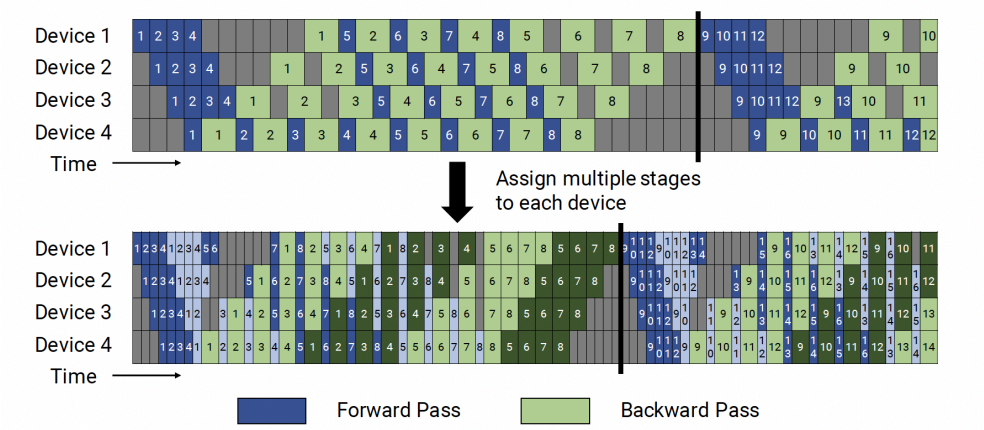

Figure 2. Default and interleaved 1F1B pipeline model parallelism schedules

Tensor model parallelism. Large matrix computations are distributed across multiple GPUs by partitioning weight and activation tensors for parallel processing. The scatter/gather method with InfiniBand and NVLink scatters data for concurrent computation and gathers component outputs to form the output tensor, enabling training of models that exceed individual GPU memory limits.

Data parallelism. Mini-batches are distributed across GPUs, with each GPU holding a complete copy of the model and computing gradients independently. Gradients are synchronized using all-reduce operations after each training iteration, ensuring uniform model weights across GPUs. To reduce communication overhead, techniques like gradient accumulation and optimized high-bandwidth interconnects are employed.

Results. Evaluations demonstrated remarkable efficiency and scalability in large-scale model training. A 530 billion parameter model was trained with a throughput of 15.1 petaFLOPs per second on 3072 NVIDIA A100 GPUs, achieving near-linear scaling and 52% of the GPUs' theoretical peak performance. PTD-P outperforms baseline model parallel approaches, delivering 30% higher throughput for a 175 billion-parameter model and matching model prediction quality across various NLP benchmarks. PTD-P estimates an end-to-end training time of ~3 months for a trillion parameter model.

Limitations

Several of *Pathways’* design choices and optimizations, such as gang-scheduling and parallel asynchronous dispatch, are primarily designed for TPUs and don’t hold for GPUs. The system’s unique contributions are blurred by its use of existing systems and techniques. Comprehensive

evaluations are lacking, notably in direct comparisons with systems like Ray and in discerning whether performance gains are due to unique architectural elements or implementation details like on-device object stores. Furthermore, while *Pathways* excels on large-scale workloads, its performance diminishes on small-scale workloads due to overheads.

Megatron-LM is implemented and evaluated on NVIDIA A100 GPUs. Crucially, the 3D parallelism strategies in *Megatron-LM* assume GPUs under ideal, reliable-network, no-failure conditions and do not address fault-tolerance. Furthermore, the paper assumes that the GPU servers are homogenous, which is idealistic under a cost-constrained environment. Moreover, its focus on transformer models may limit its applicability to other more diverse ML tasks.

Future Research Directions

Pathways specifically is optimized for Google TPUs. While such optimizations are typical and often necessary to achieve peak performance, an area of future work is generalizing certain techniques to other, potentially heterogeneous, accelerators.

Additionally, both papers exhibit a significant gap in understanding the impact of communication overhead in distributed machine learning systems, especially in complex network topologies spanning multiple data centers. There ought to be further rigorous quantitative analysis to measure how data flows across such intricate, geographically-distributed networks that make up real-world deployment scenarios.

Summary of Class Discussion

Assumptions made in GenAI Systems papers. The discussion began with questions regarding the system designs' impact on model inference performance and activation recomputation's role in improving performance. These points brought up an important discussion on how many GenAI systems papers presuppose perfect, non-failing systems, assumptions recognized as significant limitations. We also explored assumptions regarding the independence between systems and models. Often, it is assumed that strictly architectural changes will not affect models' inference accuracy and LLM semantics/standards should be maintained, despite potential systems optimizations requiring such changes.

Fault-tolerance and limitations in the Pathways and Megatron-LM systems. We delved deeper into fault-tolerance, an aspect not extensively covered in the papers. We debated the idea of using GPU "pipeline bubble" time in the *Megatron-LM* paper for redundant computations to improve fault-tolerance, balanced against the potential drawbacks of increased memory pressure and reduced throughput. Alternatively, we discussed fallback strategies involving reverting to the last checkpoint before a failure and reallocating devices, referencing Amazon's research on in-memory checkpointing to avoid performance losses associated with disk storage.

Appendix - Additional Class Questions

Evaluation

- “The capability of Pathways to retain state-of-the-art performance for current models while enabling exploration of new systems and ML research ideas is impressive. A suggestion I have for Pathways is to be compared with similar orchestration layers, exploring key differences, strengths, and shortcomings in a comparative manner. Or is this already explored?” - *Anonymous*
- “For pipeline-parallel models, how did the performance of Pathways compare with existing multicontroller systems that had implemented their own coordination primitives?” - *Anonymous*
- “There's impressive achievement [in Megatron-LM] with 1 trillion model parameters at 502 petaFLOP/s on 3072 GPUs and 52% per-GPU throughput. However, what considerations or trade-offs led to the decision not to improve per-GPU throughput further? Does this optimization level have any specific limitations or potential bottlenecks for future improvements?” - *Anonymous*

Limitations

- “Considering the proposed approach [in Pathways], is there any consideration or solution regarding the potential privacy or security concerns that may arise during the parallel execution on distributed compute devices?” - *Anonymous*
- “Deploying and running large-scale ML models using PATHWAYS and TPU islands might be economically prohibitive for smaller research teams or institutions. How do you envision democratizing access to such high-performance ML training capabilities, and are there any plans to address this within the PATHWAYS project?” - *Reviewer Team (Chris, Aiden, Maya)*
- “Given the substantial reliance on NVLink and NVSwitch technologies for the [Megatron-LM] system's performance, how adaptable is PTD-P to environments where such interconnects are not available or are of lower bandwidth? Would this significantly degrade performance, or are there fallback strategies in place?” - *Reviewer Team (Chris, Aiden, Maya)*

Future Work

- “The parallel dispatch method [in Pathways], while fast with static work, can fall short when the computation is data-dependent (e.g. in MoE models). What are some research paths that could optimize this?” - *Anonymous*
- “Large-scale ML training operations are energy-intensive. Can you discuss any evaluations of PATHWAYS in terms of energy efficiency and its potential to reduce the environmental impact of large-scale ML computations?” - *Reviewer Team (Chris, Aiden, Maya)*
- “The research predominantly focuses on the GPT model architecture. How applicable or easily adaptable is the PTD-P system to other deep learning models, especially those with significantly different architectural or computational characteristics?” - *Reviewer Team (Chris, Aiden, Maya)*