Pathways: Asynchronous Distributed Dataflow for ML

by Barham et al., 2022

Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Dan Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, Brennan Saeta, Parker Schuh, Ryan Sepassi, Laurent El Shafey, Chandramohan A. Thekkath, Yonghui Wu

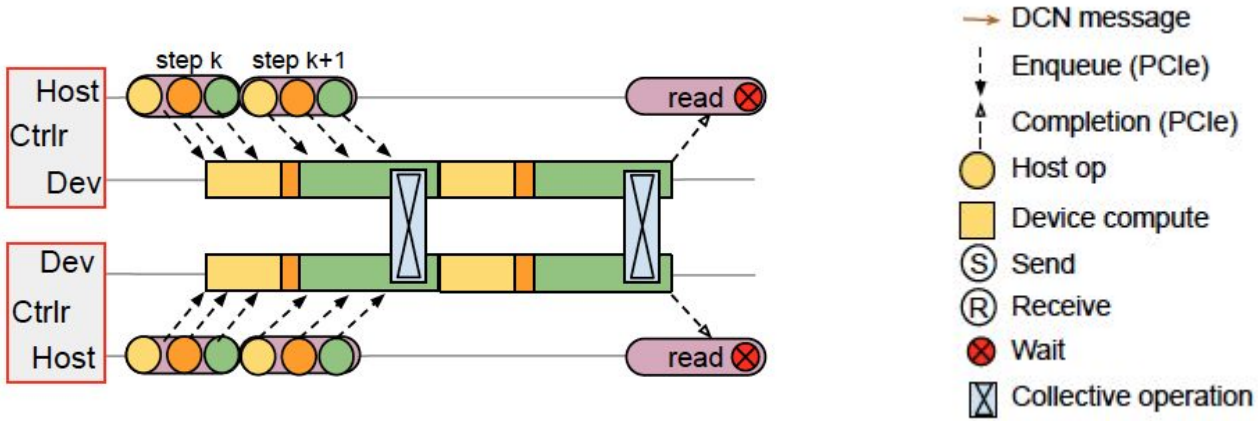
[Paper link]

Presented by Andrzej Szablewski, as3623@cam.ac.uk

23/10/2024

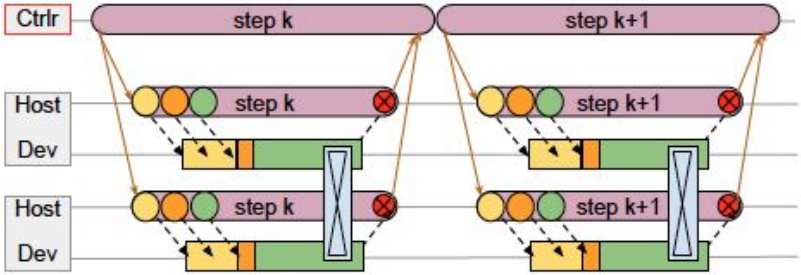
Yesterday…

- Single Program Multiple Data paradigm (SPMD)
  - All accelerators run the same computation and communicate collectively!

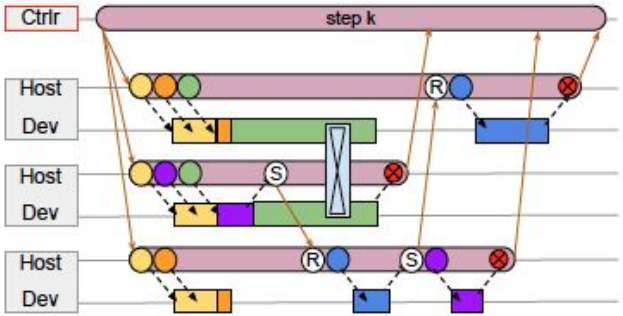


(a) JAX/PyTorch SPMD

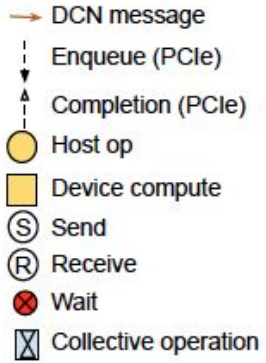
Yesterday…



(b) TF1 SPMD



(c) TF1 non-SPMD



Yesterday… and Today!

- Single Program Multiple Data paradigm (SPMD)
  - All accelerators run the same computation and communicate collectively!

- Large models struggle to scale using SPMD. Instead:
  - Pipelining
  - Use of sparsity (e.g. MoE)
  - …
- Heterogeneous clusters
  - Multiple Program Multiple Data (MPMD)
- Foundation models and model sharing

SPMD models and multi-controller (PyTorch, Tensorflow, JAX)

- Low latency (dispatch over fast PCIe) 🔥
- Implementation fairly straightforward (ctrl+c, ctrl+v) 🔥
- but…
- Poor flexibility! 😖

So, maybe single-controller systems?

- General distributed dataflow model (MPMD) 🔥
- Virtualisation of resources 🔥
- but…
- Dispatch over slower Data Center Network (DCN) 😖
- Still important to have SPMD ability -> Gang-scheduling 😖

The Challenge

Low latency + workload flexibility

Solution!

Sharded dataflow graph + Asynchronous operations

Pathways Programming Model

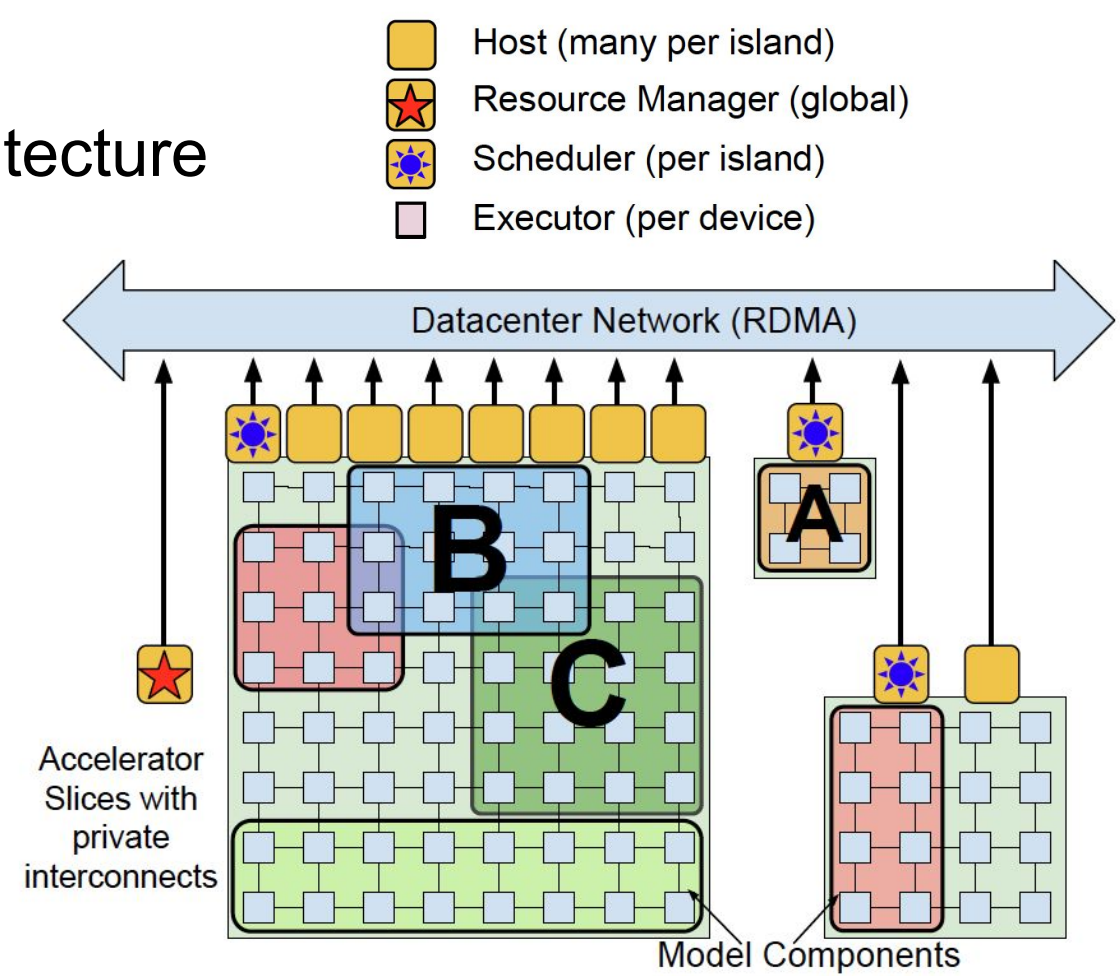
- XLA computations -- “compiled functions”
  - Resources known beforehand (input/output shapes, types, etc.)
  - Each compiled function maps to a single (sharded) node in the dataflow graph
- Functions can be placed on specific virtual devices with desired network topology and other constraints
- Pathways automatically handles data movement and resharding

Pathways System Architecture

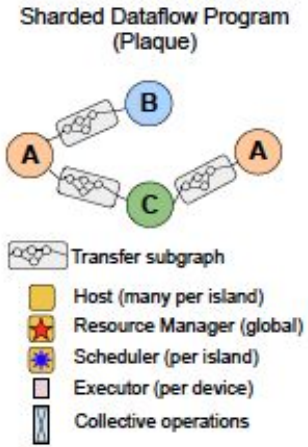
- Resource Manager
- Client
- Coordination
- Gang-scheduled dynamic dispatch
- Parallel async dispatch
- Data management

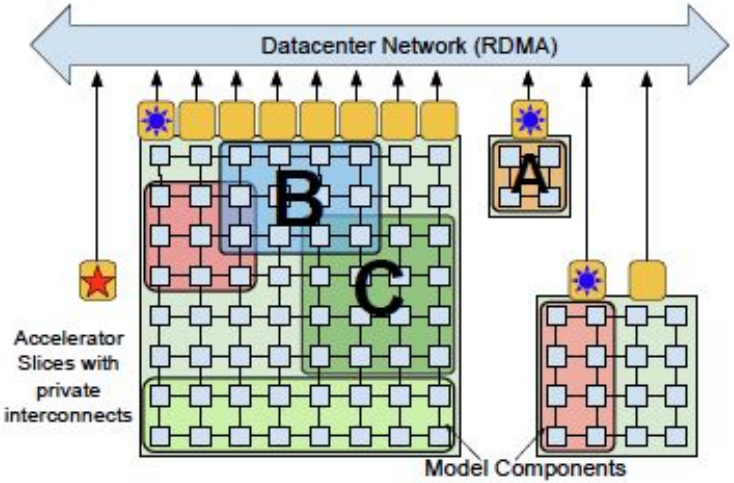
Pathways System Architecture

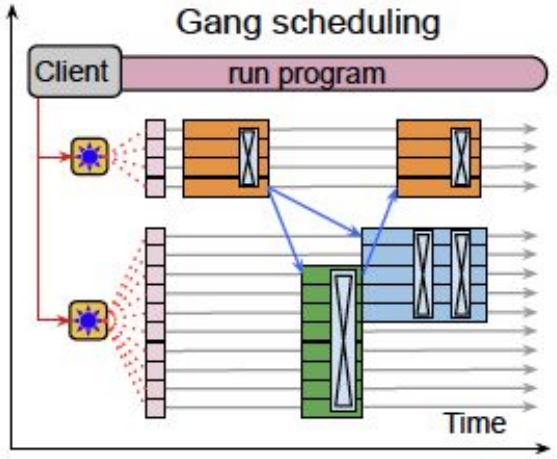
- Resource Manager
- Client
- Coordination
- Gang-scheduled dynamic dispatch
- Parallel async dispatch
- Data management



Pathways System Architecture

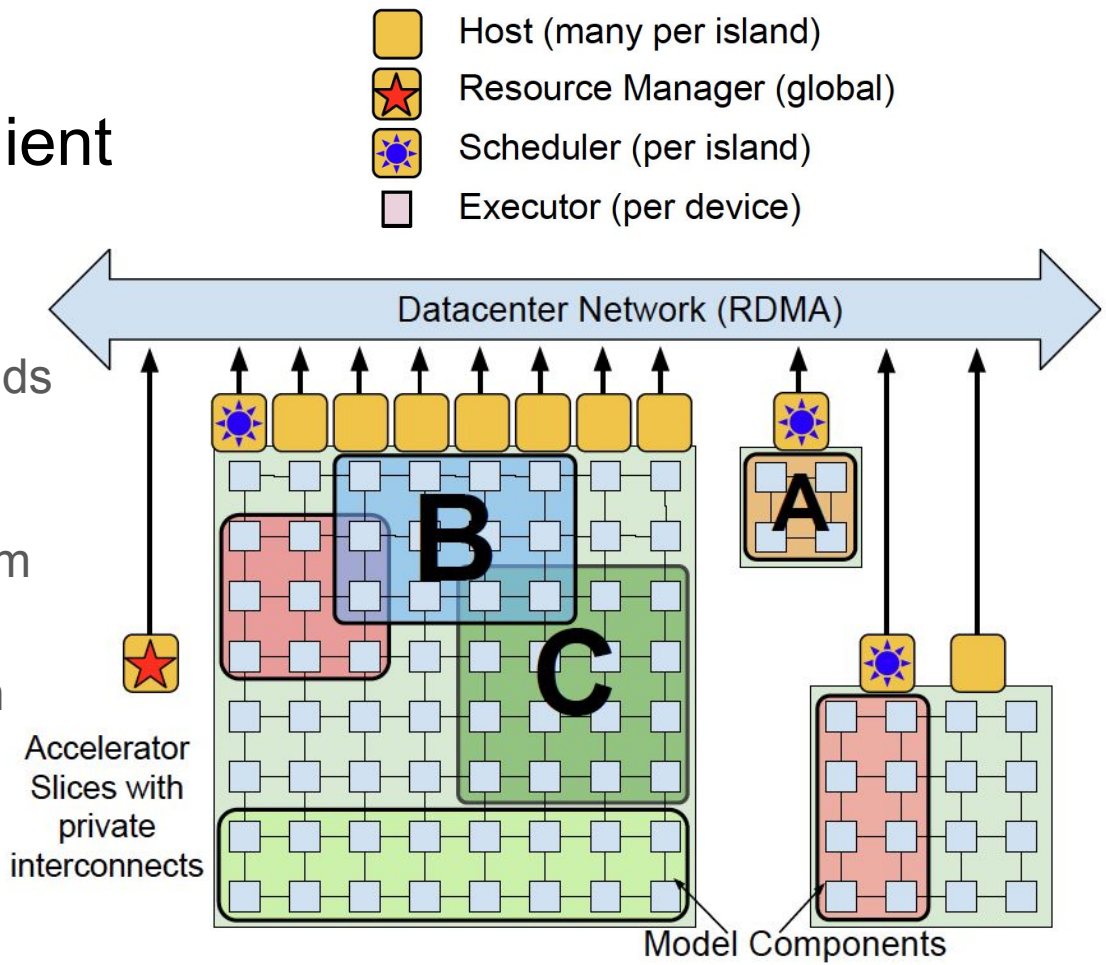






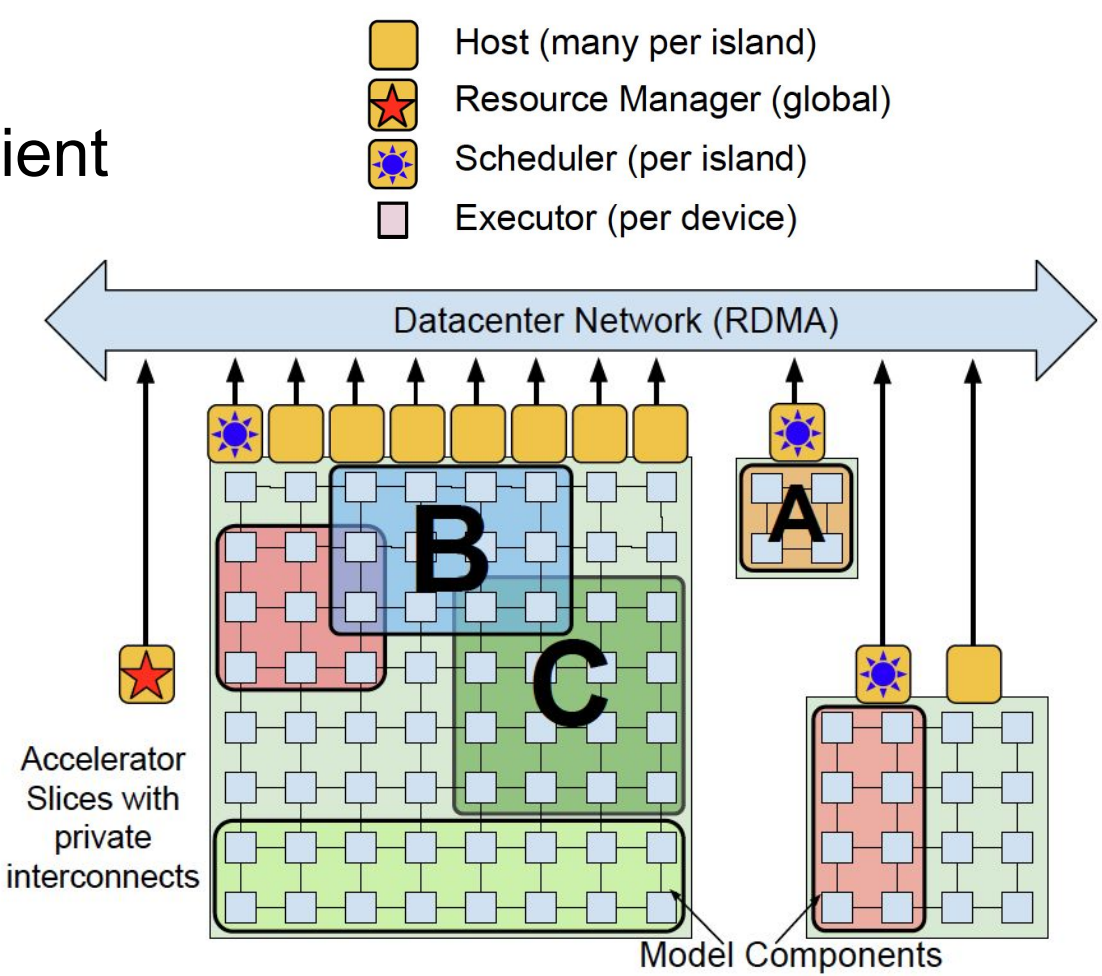
Resource Manager & Client

- Accelerators grouped in islands
- Virtualisation of resources
  - one-to-one virtual to physical
- Dynamic scaling of the system
- Single-controller allows pause/resume and migration



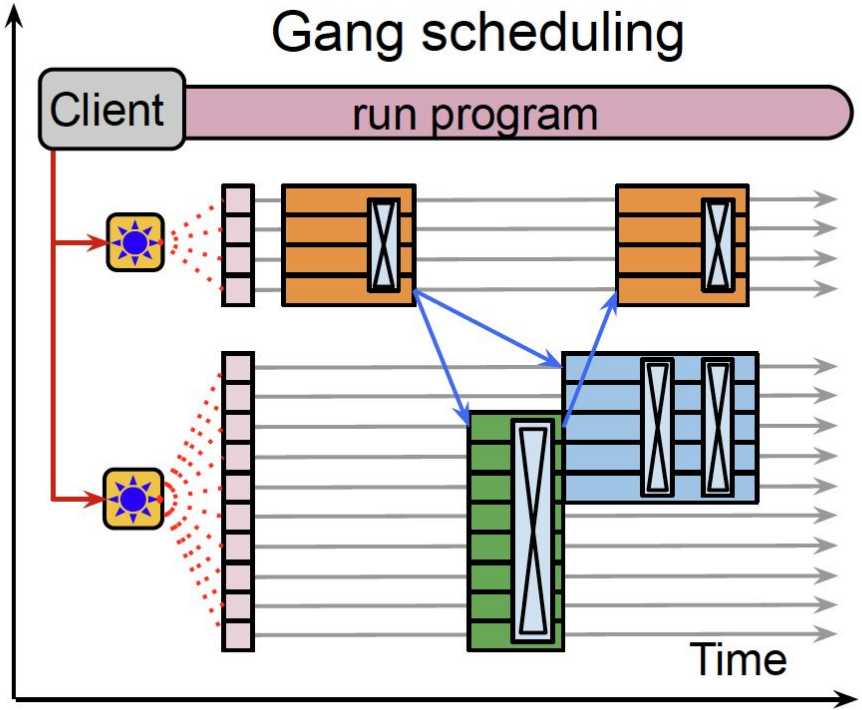
Resource Manager & Client

- Client registers computations and get them compiled
- Creates device-agnostic IR
- Sharded buffer for instruction and data buffers

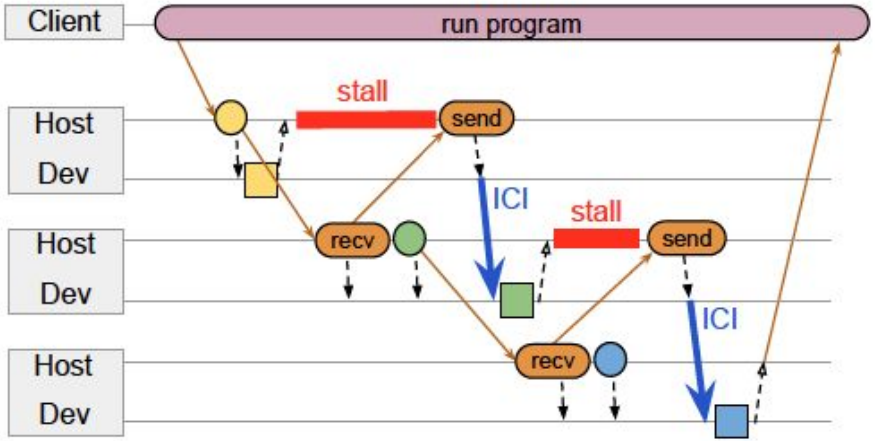


Coordination and Gang-Scheduling

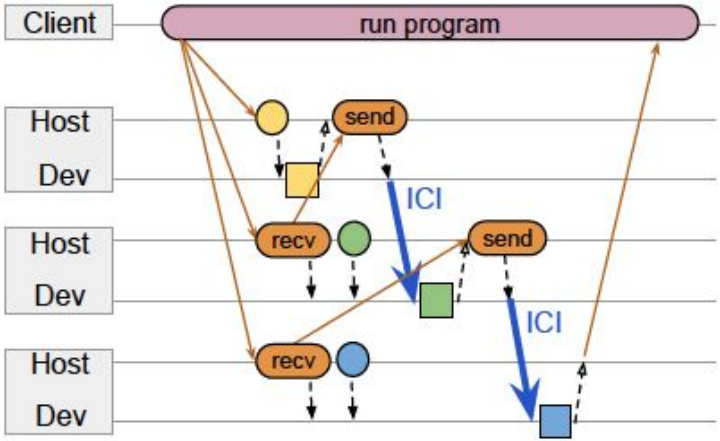
- Cross-host coordination using *Plaque* -- closed-source (😡) sharded dataflow system
  - Sparse, low-latency communication
- Plaque
  - Async enquing the execution of computation and network sends
  - Communication with scheduler for consistent ordering of fn execution
- Centralised scheduler per island managing all computation
  - Simple FIFO allocation



Parallel Asynchronous Dispatch



(a) Sequential dispatch



(b) Parallel dispatch

Evaluation

- Single-controller dispatch overheads
  - set of mini-benchmarks
- Multi-tenancy
- Large scale model performance (real machine learning workloads)
  - text-to-text transformer
  - large language model

Single-Controller Dispatch Overheads

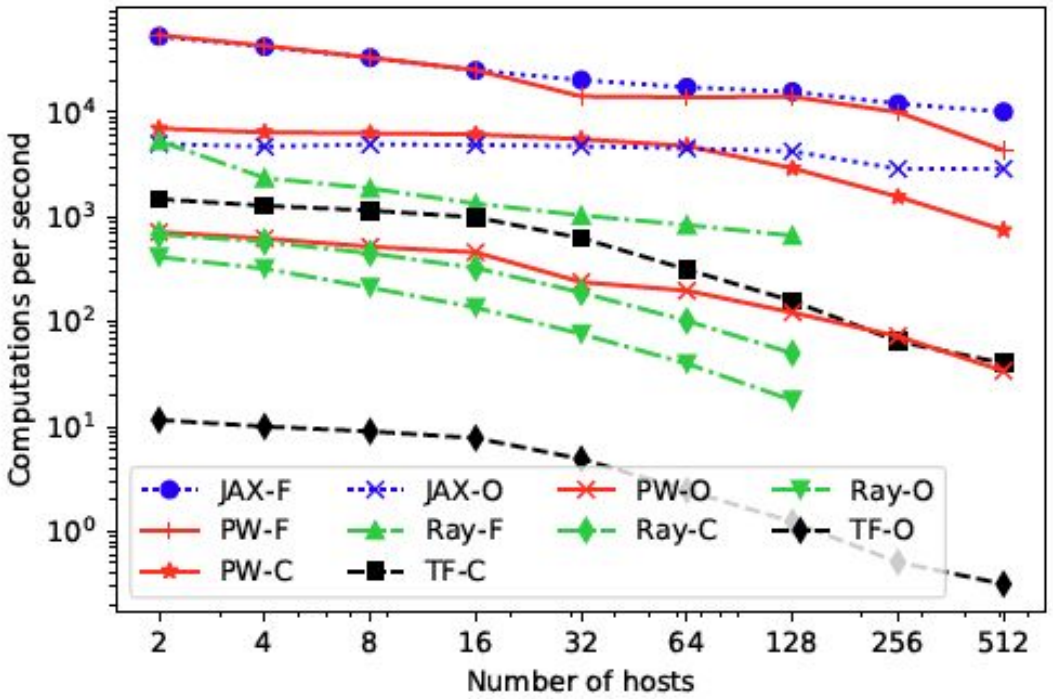
Trivial gang-scheduled computation containing a single AllReduce.

Measuring throughput

-O - Separate call for each instruction

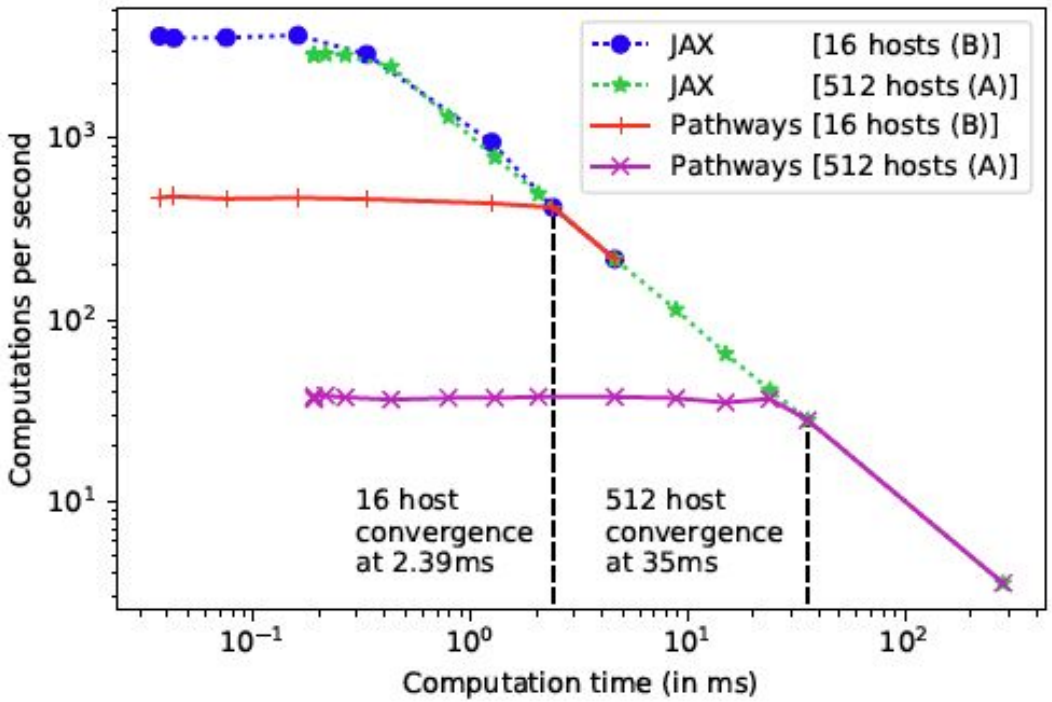
-C - Chained, 128 nodes,
single client call

-F - Fused, single node
single client call



Single-Controller Dispatch Overheads

Smallest computation to match throughput between Pathways and JAX, masking the single-controller overhead.



Multi-Tenancy

Aggregated throughput when multiple clients concurrently submit different Pathways programs

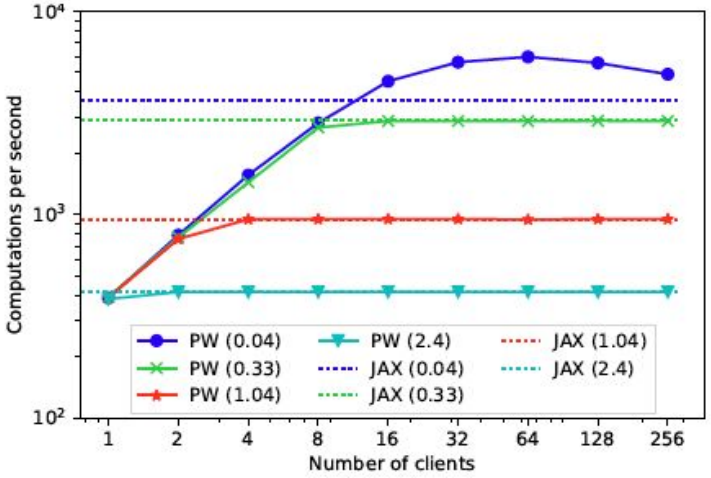


Figure 8. Aggregate throughput of concurrent programs (compute times in ms). PATHWAYS time-multiplexes accelerators between programs efficiently incurring no overhead to context switch.

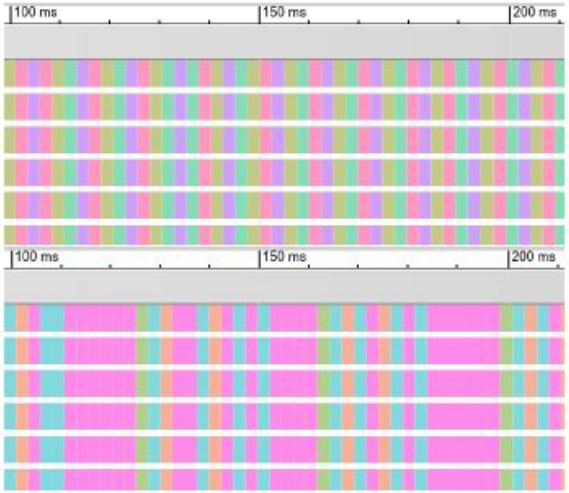


Figure 9. Traces of a sample of cores on PATHWAYS showing interleaving of gang-scheduled concurrent programs with proportional-share ratios of 1:1:1:1 (Upper) and 1:2:4:8 (Lower) between 4 clients.

Large Scale Model Performance

Text-to-text Transformer

Identical performance because realistic computations are large enough to mask single-controller overheads!

Table 1. Training throughput (tokens/s) of Text-to-text Transformer model configurations from (Raffel et al., 2019) on JAX multi-controller and PATHWAYS.

Model	Params	TPU cores	JAX	PATHWAYS
T5-Base	270M	32	618k	618k
T5-Large	770M	32	90.4k	90.4k
T5-3B	3B	512	282.8k	282.8k
T5-11B	11B	512	84.8k	84.8k

Large Scale Model Performance

3B Transformer-based language model (decoder-only)

Table 2. Training throughput (tokens/s) of 3B Transformer language model, using SPMD or multiple pipeline stages, with C TPU cores in PATHWAYS. For pipeline-parallel models, there are S stages and each batch is split into M μ-batches.

Model configuration	TPU cores	PATHWAYS
Model-parallel (SPMD)	128	125.7k
Pipelining, S=4, M=16	128	133.7k
Pipelining, S=8, M=32	128	132.7k
Pipelining, S=16, M=64	128	131.4k
Pipelining, S=16, M=64	512	507.8k

Large Scale Model Performance

3B Transformer-based language model (decoder-only)

Here pipeline has competitive performance to SPMD.

Collective communication within the SPMD computation incurs higher overhead than pipeline bubble overhead.

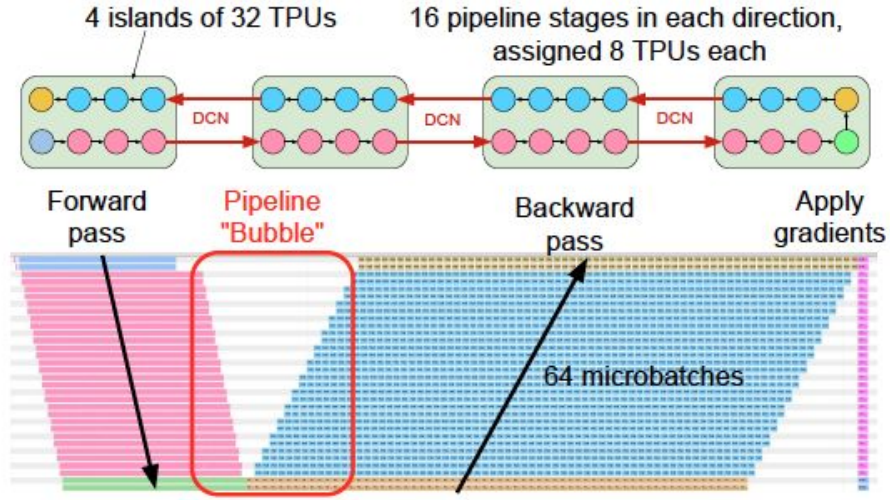


Figure 10. 3B Transformer model pipelined over 128 TPUs: PATHWAYS can efficiently train models over islands of TPUs connected via DCN achieving the same throughput (131.4k tokens/sec) on 4 islands of 32 cores each on configuration (C) as using a single island of 128 cores on configuration (B).

Final Thoughts and Further Research

- Design impacted by the use of TPUs over GPUs -- fusing many computations into a TPU kernel.
- No software released and use of closed-source systems 😖

- Room for further improvements in dynamic resource management
- Fine-grained control flow -- selective parameter updates
  - Useful for MoE architectures with routing
  - Data-dependent data exchanges between nodes

Good news!

- Single-controller model allows simple access to much richer computation patterns
- Low-latency achieved with minimal overheads in real machine learning scenarios

Thank you!