Pathways: Asynchronous Distributed Dataflow for ML

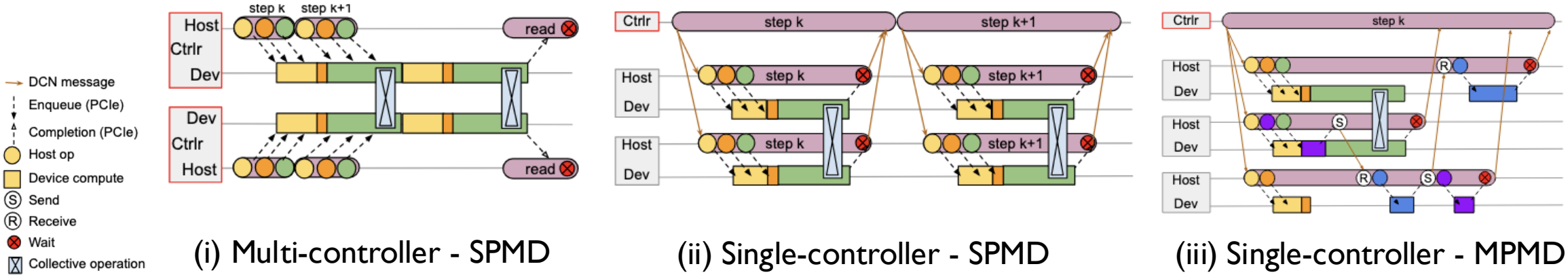
Based on Barham et al., MLSys 2022

Presented by Yavuz Ferhatosmanoglu

Parallel Execution and Control Models

- Execution Models
  1. Single Program Multiple Data (SPMD)
  2. Multiple Program Multiple Data (MPMD)
     - Enables the use of heterogenous computation and clusters, model sparsity, and pipeline/model parallelism
- Control Models
  1. Single-controller
     - Dispatch of computations to hosts over a DCN is high-latency
  2. Multi-controller
     - Inflexible in terms of resource virtualisation, since each host has its own exclusive controller

Existing Execution Schemes



(i) Multi-controller - SPMD (JAX/PyTorch)

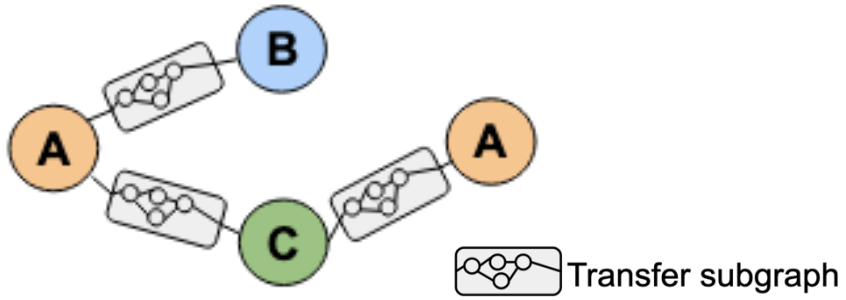
(ii) Single-controller - SPMD (TensorFlow v1)

(iii) Single-controller - MPMD (TensorFlow v1)

- Main contributions of Pathways:
  - Provides flexibility of single-controller systems with performance of multi-controller
  - Enables efficient future research for MPMD, while maintaining state-of-the-art performance for current SPMD use-cases

Pathways Dataflow Model

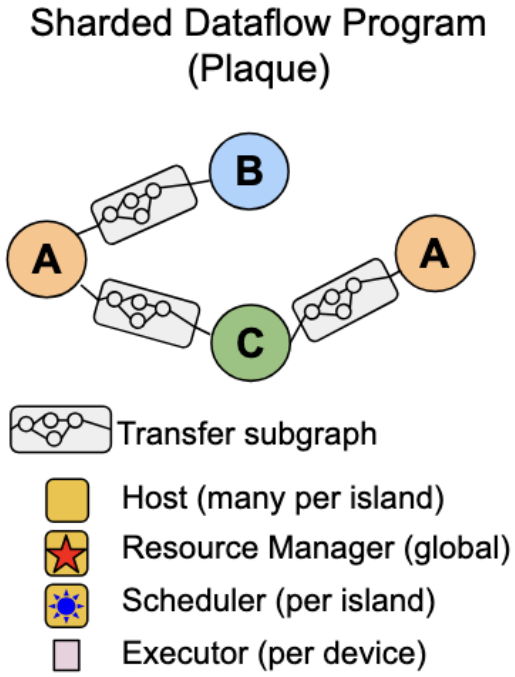
- **Sharded** dataflow graph consisting of asynchronous operators
- Nodes in the graph are compiled functions, to be placed on virtual devices satisfying network topology and memory constraints
- Can be used as a drop-in replacement for the JAX backend
  - Nodes represent XLA computations
  - Enables existing JAX code to scale beyond a single TPU pod

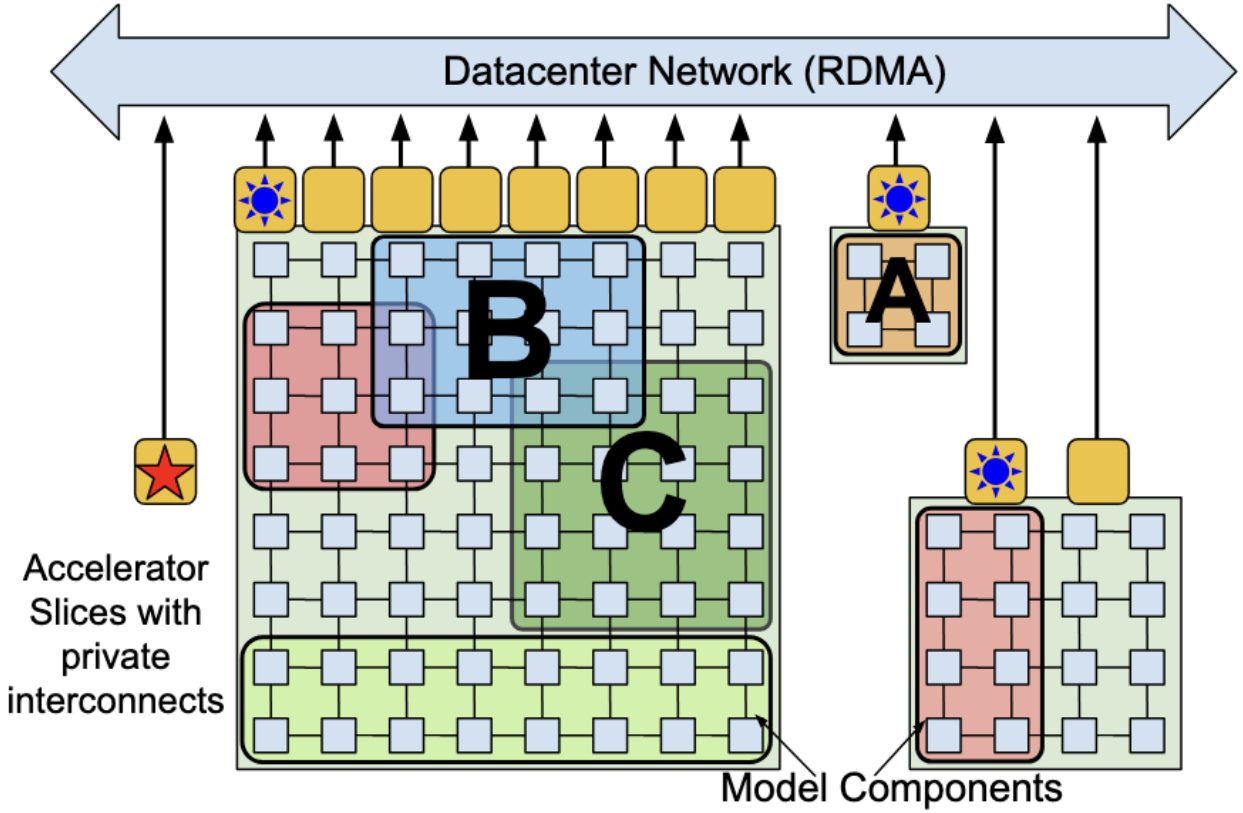


Sharded Dataflow Program

Pathways System Architecture

- Backend: set of accelerators grouped into tightly-coupled islands
- Resource manager: centralised management of devices, enabling virtualisation



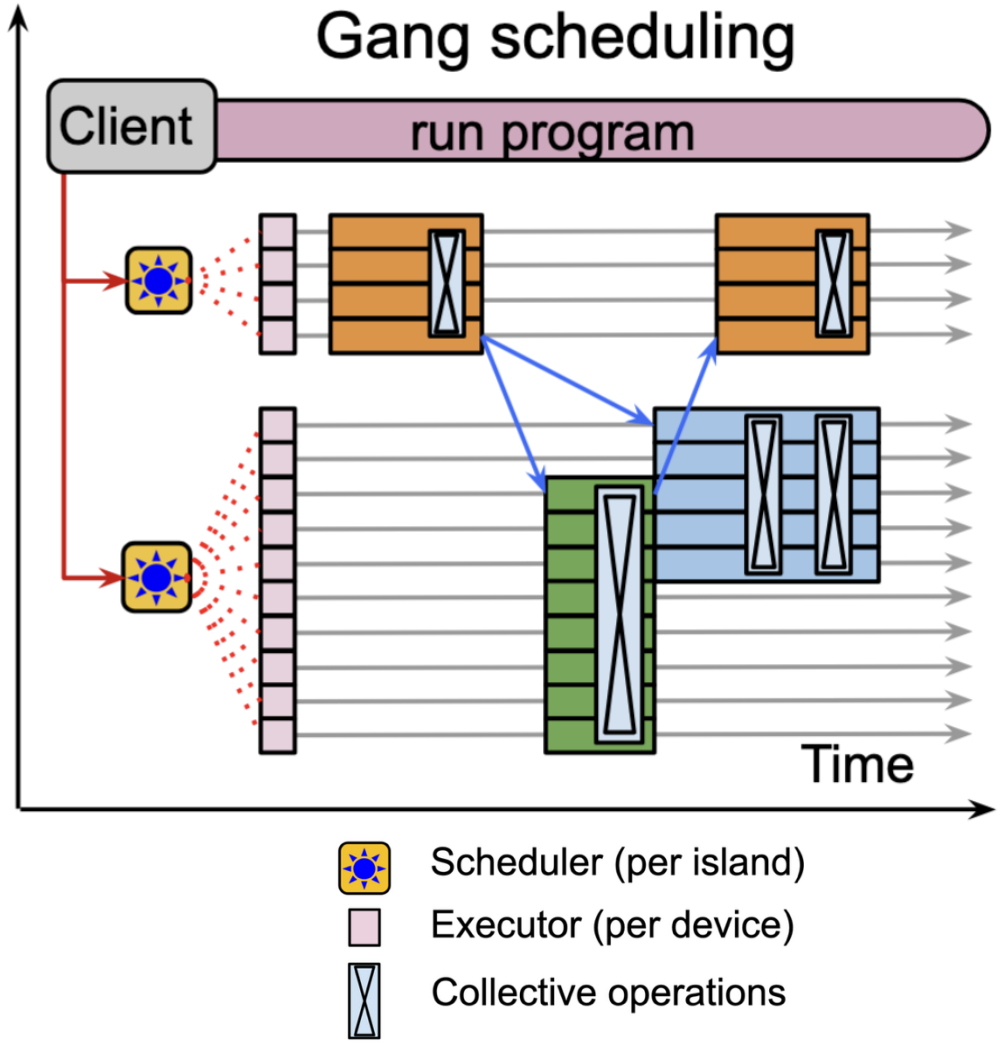


Client Operations and Coordination

- Users interact with the Pathways client:
  - Registers computations with the resource manager
  - Compiles user programs to the device location-agnostic Pathways IR
  - Uses **sharded buffer** that may be distributed over multiple devices, to avoid the client becoming the bottleneck
- Cross-host coordination uses Google's closed-source *Plaque*:
  - The low-level Pathways IR is converted into a *Plaque* program
  - Provides low-latency critical messaging over a DCN
  - Batches messages to the same host for high-throughput requirements

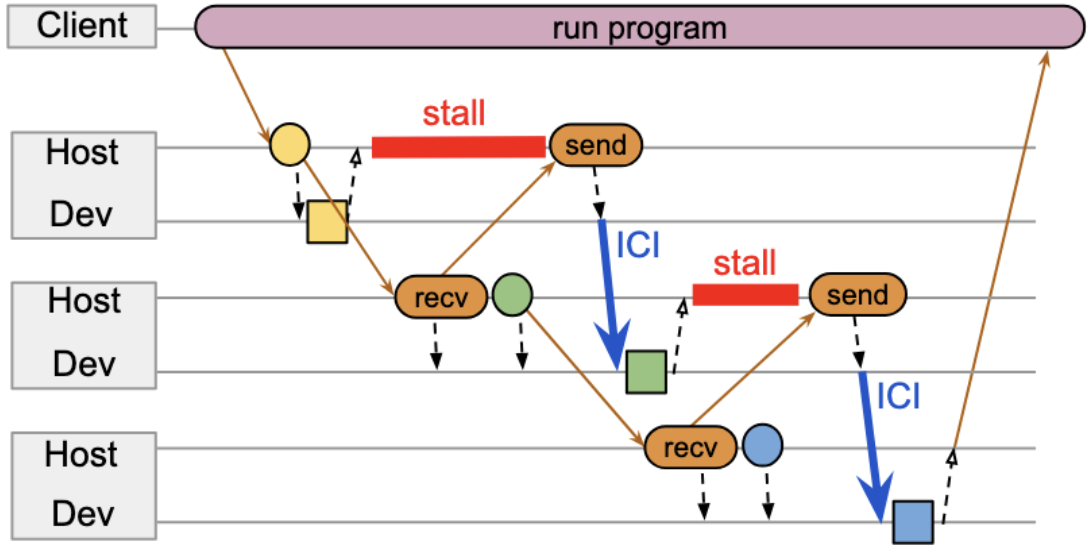
Gang-scheduled Dynamic Dispatch

- Per-island centralised schedulers order computations across devices
- *Plaque* programs:
  1. Enqueue execution of node with buffer futures as input.
  2. Enqueue network sends for the buffer futures output by a node.
  3. Communicate with the scheduler to determine a consistent order of node executions in that island.

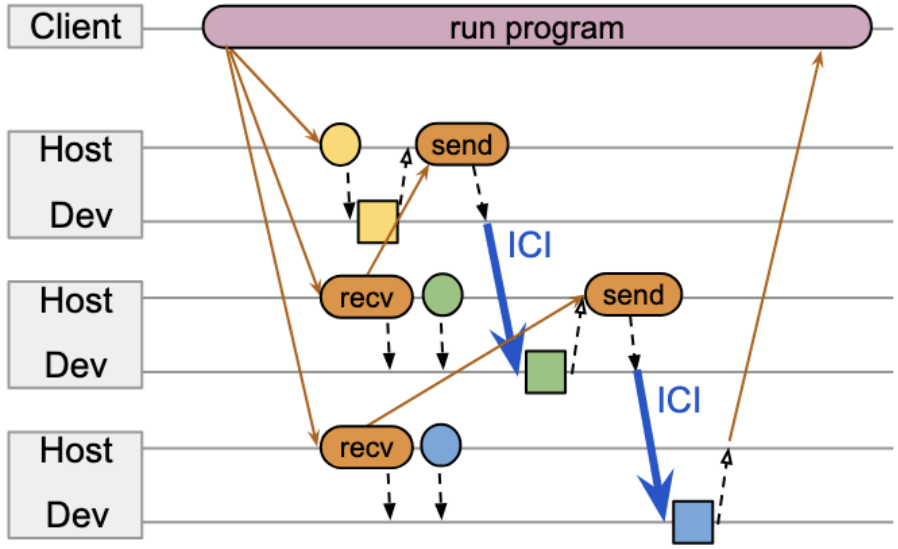


Parallel Asynchronous Dispatch

- Run the host-side work (scheduling, resource allocation and coordination) in parallel to device computations to reduce time spent stalling
- This can only be done if the compiled functions are regular



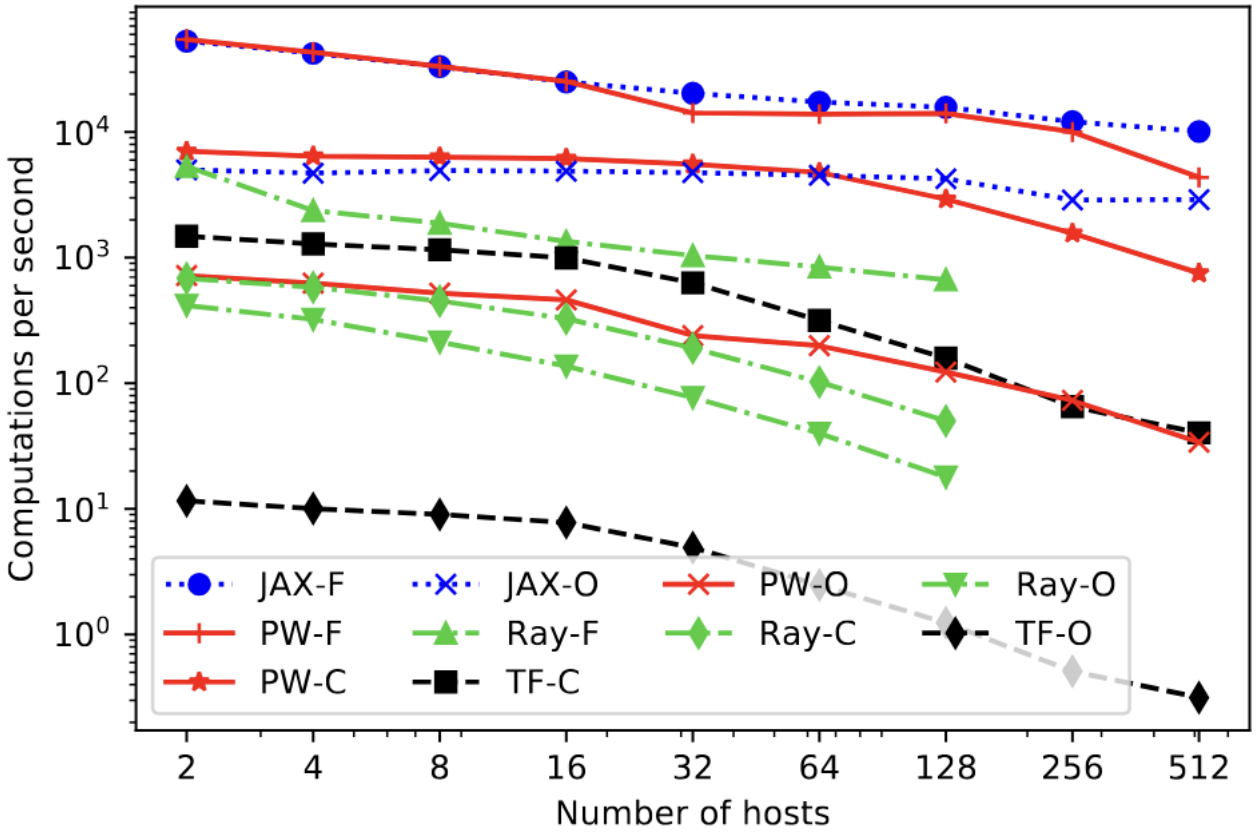
(a) Sequential dispatch



(b) Parallel dispatch

Evaluation: Dispatch Overheads

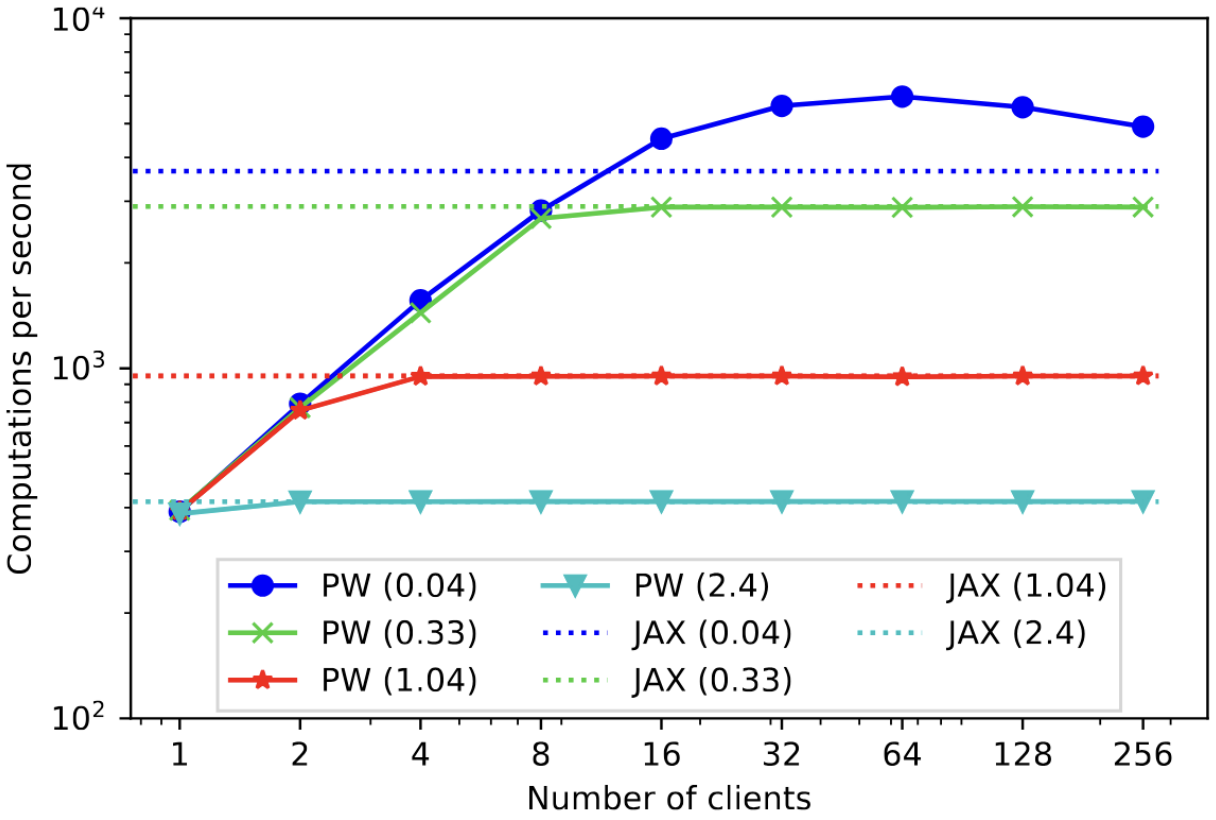
- Micro Benchmark: trivial gang-scheduled computation containing a single AllReduce operation
- Three settings:
  - OpByOp (–O): user code contains a separate call for each execution
  - Chained (-C): a series of calls, each executing 128 nodes
  - Fused (-F): a series of calls, each executing a single node, but each node containing 128 computations



Pathways outperforms single-controller systems like TF and Ray, and matches multi-controller JAX in -F and -C configurations for up to 1000 and 256 TPU cores, respectively.

Evaluation: Multi-tenancy

- Analysis of aggregated throughput when multiple clients concurrently submit different Pathways programs.
- Matches performance of JAX as the number of clients increase.



Aggregate throughput for each system (compute time in ms)

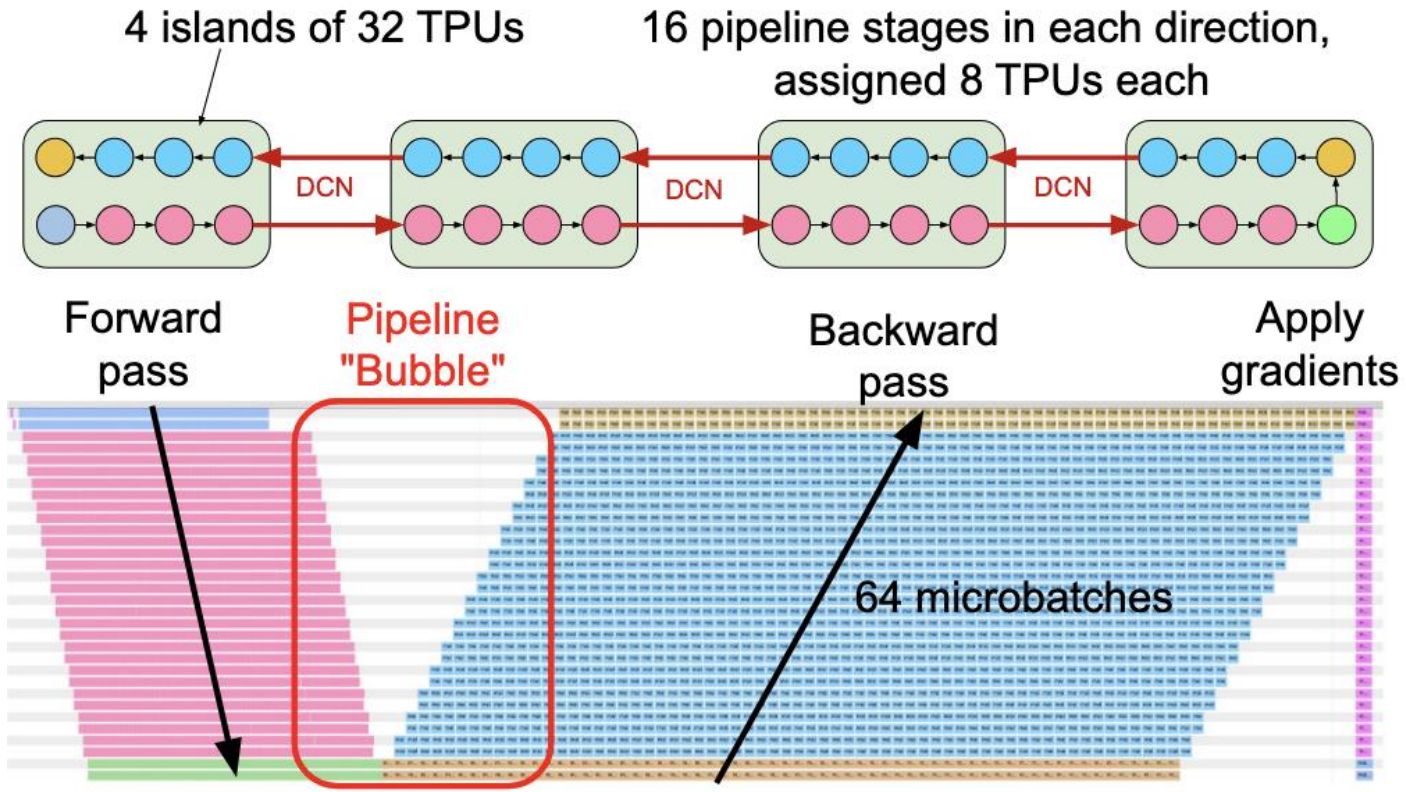
Evaluation: Large Scale Model Performance

- Training real ML models as SPMD programs
- For Text-to-Text Encoder-Decoder Transformer models, throughput (tokens/s) of JAX and Pathways is **identical**
  - Computations are large enough to mask single-controller overheads.

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

Evaluation: Large Scale Model Performance

- Pipelining the training of a Transformer model (3B parameters)
- Competitive throughput (131.4k tokens/s) can be achieved compared to SPMD.



Evaluation: Large Scale Model Performance

- Comparison of SPMD and Pipelined performance for Decoder-only Transformer model
- Training throughput given number of stages (S) and micro-batches (M)
- Pipelined performance can be better than SPMD, and throughput scales linearly with number of hosts

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

Final Thoughts: Positives

- Matches state-of-the-art performance, while enabling future research directions
- Drop-in replacement for the backend of existing JAX code
- Provides great flexibility, enabling future research directions in ML
- Enables large-scale use of (internal) large-scale models such as Google's PaLM (540B)
- Core part of Google's "AI Hypercomputer" (2025)
  - Pathways is now available as part of Google Cloud for inference and training

Final Thoughts: Criticisms

- Architecture is only for TPUs as the execution backend
  - Authors only speculate that the system can be adapted to use GPUs
- Relies on Google's closed-source *Plaque*
  - Authors again only speculate that *Ray* can be used instead
- Details omitted, such as how fault tolerance is handled
  - The work is not reproducible, and further research is needed for the general case
- Experimentation could be extended to cover a wider range of end-to-end MPMD use-cases, suggesting best practices

Thank you!
Any Questions?