Pathways: Asynchronous Distributed Dataflow for ML

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

Presentation by George Barbulescu | R244 | 19/10/2022

Problem Statement

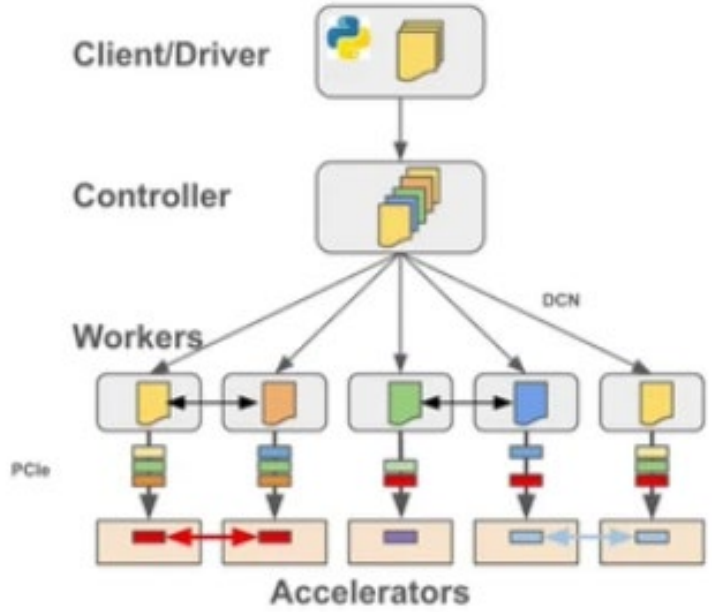
- Co-evolution across…
  - ML algorithms, especially Deep Learning
  - Hardware resources
  - Distributed ML systems
- SOTA ML orchestration software
  - Single Program Multiple Data
  - Lockstep synchronised as per MPI (Clarke et. al, 1994)
  - Prone to “overfitting” the underlying accelerator profile

Background

- Multi-controller Architecture
  - Tools such as PyTorch and JAX
  - Assume exclusive ownership of the underlying hardware profile
  - Coordination primitives are restricted
  - Robust choice for building supercomputers due to PCIe dispatch
- Single-controller Architecture
  - Spark, TF, MapReduce
  - High dispatch latency since workers communicate via DCN
  - Hard to scale up horizontally due to controller coordination overhead
  - Flexible model and virtualization opportunities

Single-Controller Architecture

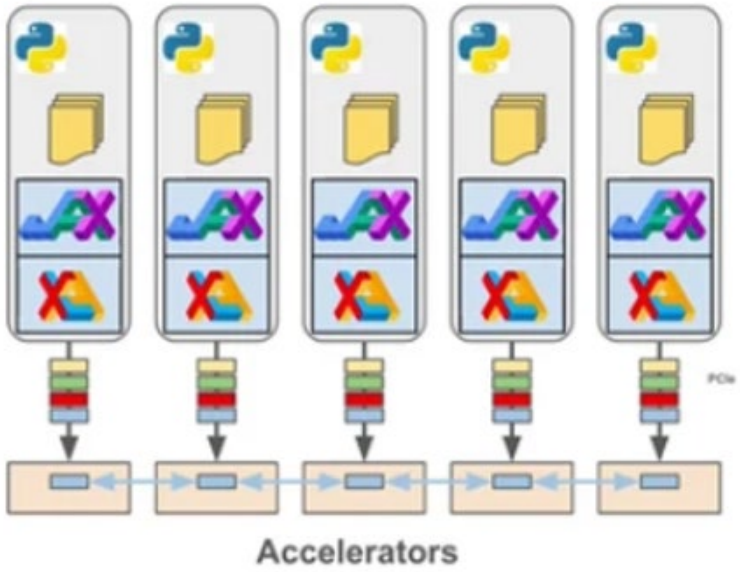
- Dataflow graph is split into subgraphs
- Work is dispatched to hosts via the datacentre network
- What happens if we add an accelerator?



Unless otherwise noted, the figures in this presentation are taken from P. Barham, et al.: Pathways: Asynchronous Distributed Dataflow for ML, MLSys, 2022. (Conference presentation/ manuscript submission to MLSys 2022)

Multi-Controller Architecture

- The client is cloned across multiple processes
- Stepwise lock based on Single Program Multiple Data
- Restrictive in terms of resource ownership and communication
- Efficient horizontal scaling!



Pathways runtime

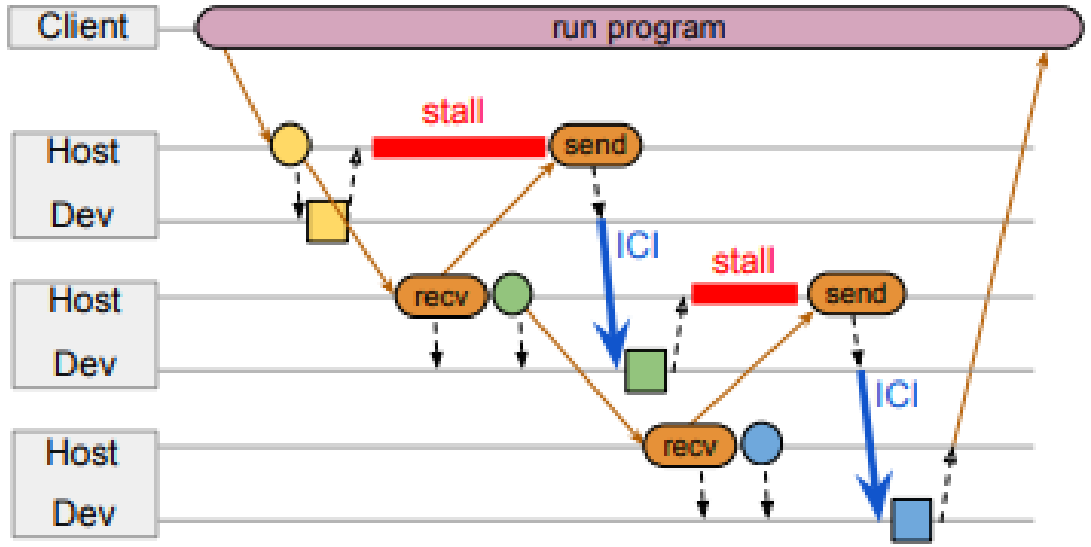
- Distributed ML runtime for cluster-wide orchestration.
  - Single-controller architecture
- Contributions
  - Sharded dataflow model with asynchronous operators
  - Decoupling between the control-plane and the data-plane
  - Asynchronous dispatch for masking single-controller latency
  - Gang-scheduling for expressive non-SPMD computations
  - Cluster-wide resource managing and virtualization for heterogenous accelerator scale-up (up to islands of TPUs!)

Insightful findings!

- Machine Learning workloads are historically “free” from complex data-driven control flow
- A high-percentage of functions classify as Accelerated Linear Algebra (XLA) computations
  - Functional data-driven conditions with predictable resource consumption for either branch
  - Bounded loops with early termination
  - Input and output types and shapes are known a priori

Sequential dispatch strategy

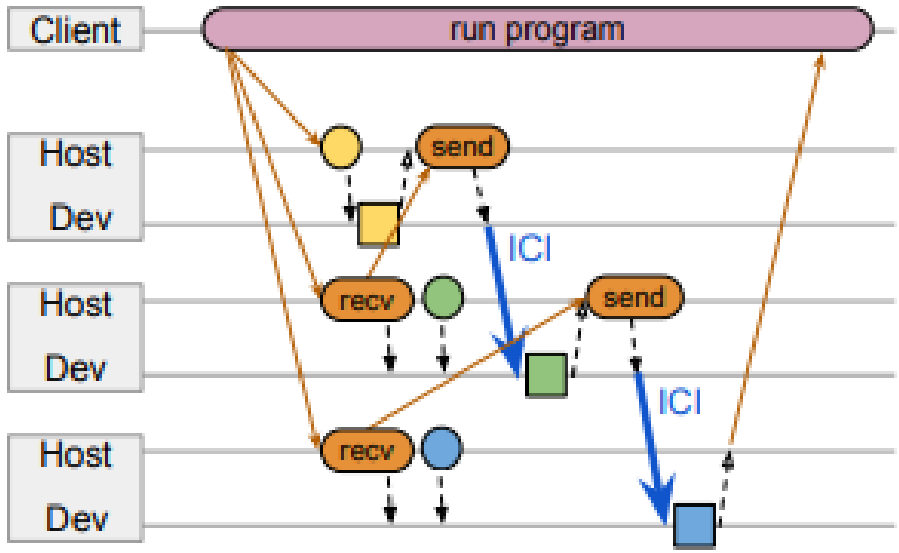
- Brute force conversion from dataflow to an asynchronous enqueuing strategy
- Reduce dispatch time over DCN but accelerators outperform queueing
- The bottleneck is in the control plane



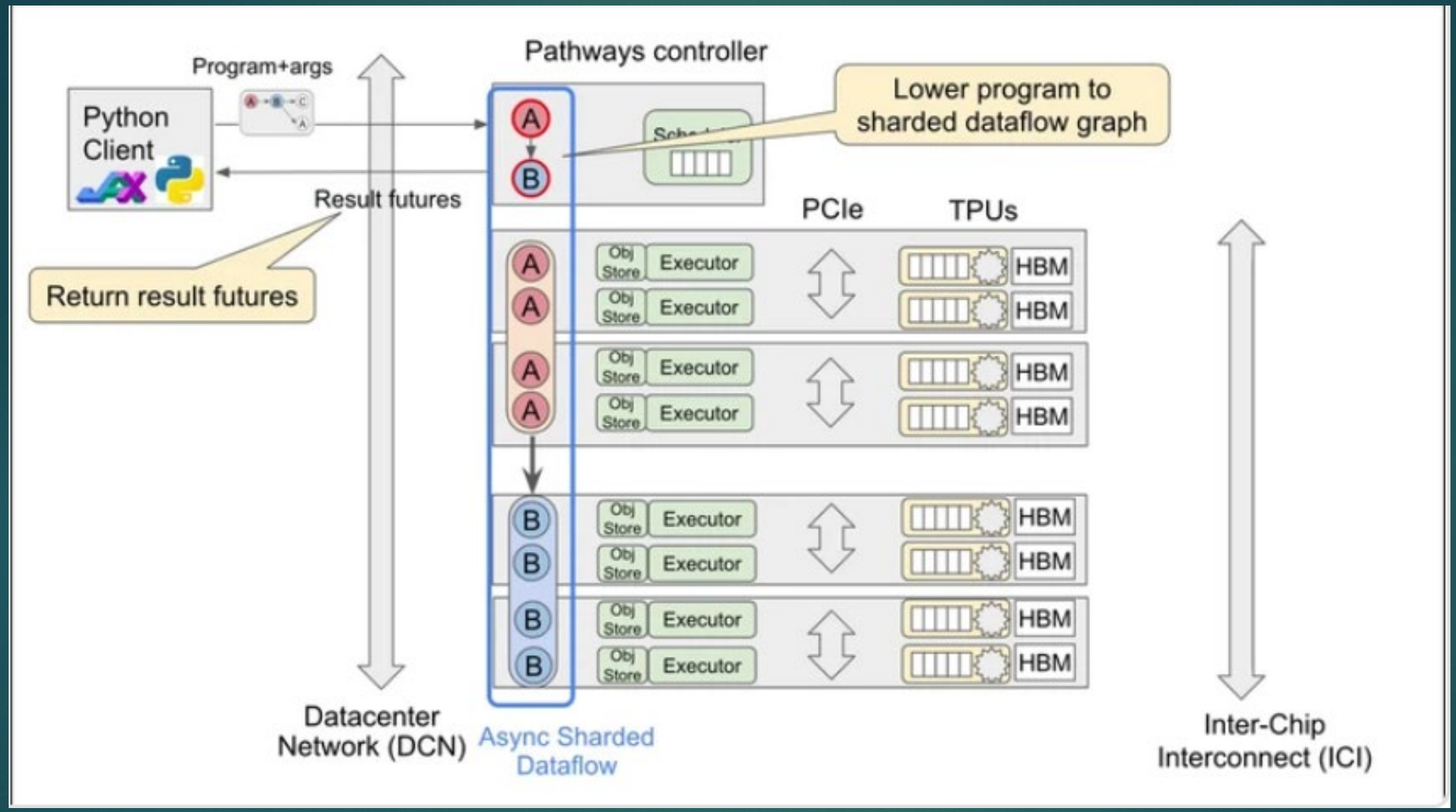
(a) Sequential dispatch

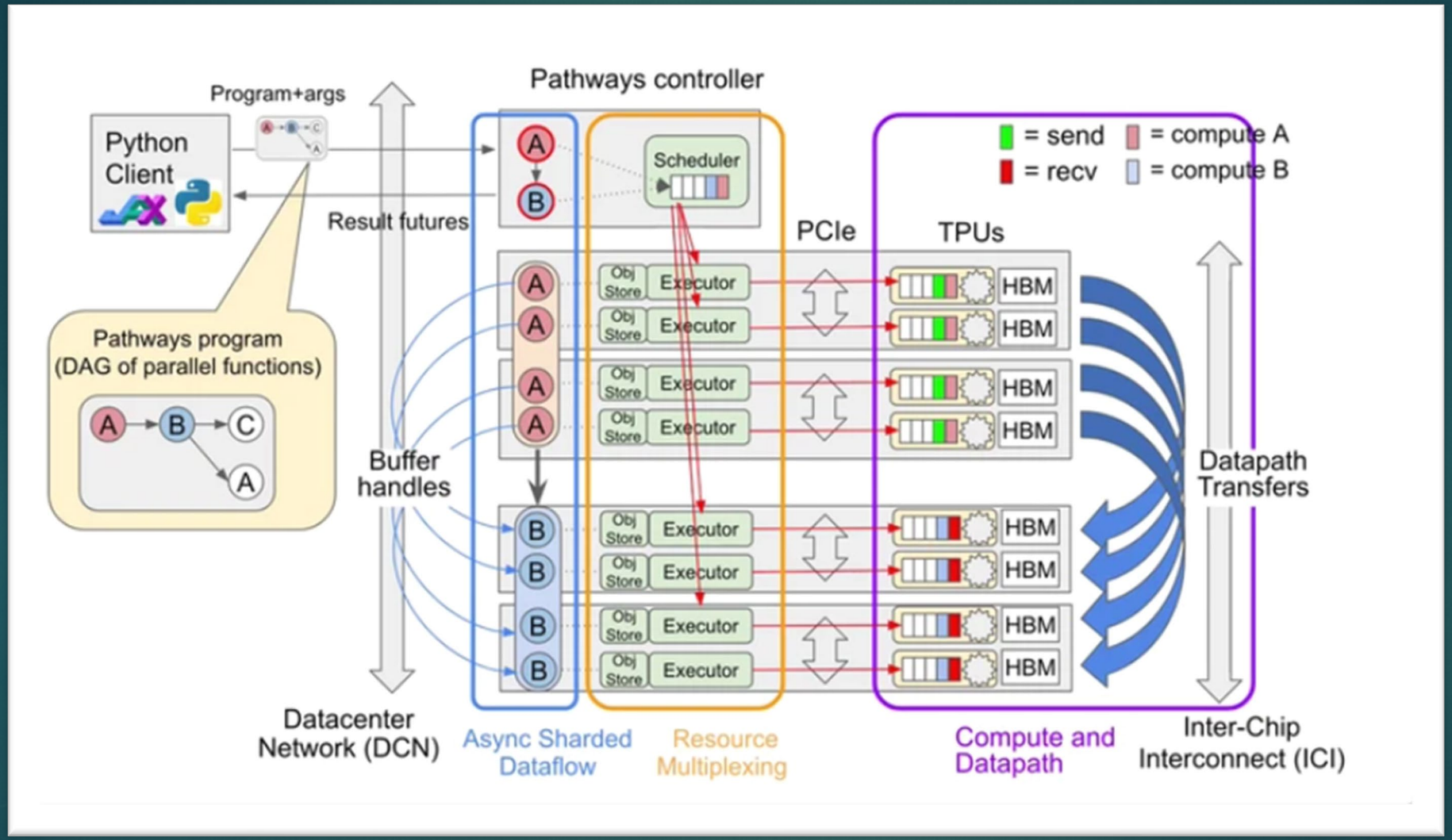
Parallel dispatch

- Control-plane latency is masked by accelerator
- Futures are sent in parallel across hosts via DCN
- Data is sent to the input buffer address after computation ends via ICI
- What happens if there's data-driven control?



(b) Parallel dispatch


Pathways controller
Program+args
Python Client
Lower program to sharded dataflow graph
Result futures
Return result futures
PCIe
TPUs
Obj Store
Executor
HBM
Datacenter Network (DCN)
Async Sharded Dataflow
Inter-Chip Interconnect (ICI)

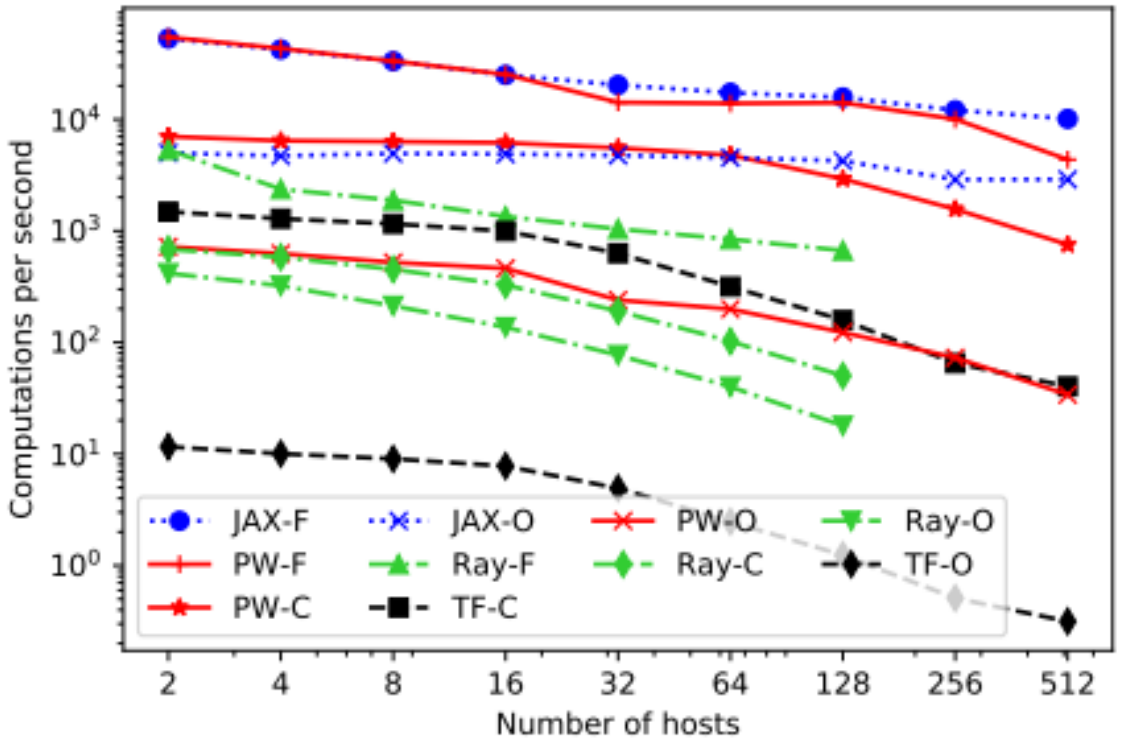

Program+args
Python Client
Pathways program (DAG of parallel functions)
A
B
C
A
Result futures
Pathways controller
Scheduler
PCIe
TPUs
= send
= recv
= compute A
= compute B
Obj Store
Executor
HBM
Buffer handles
Datapath Transfers
Datacenter Network (DCN)
Async Sharded Dataflow
Resource Multiplexing
Compute and Datapath
Inter-Chip Interconnect (ICI)

Implementation for JAX

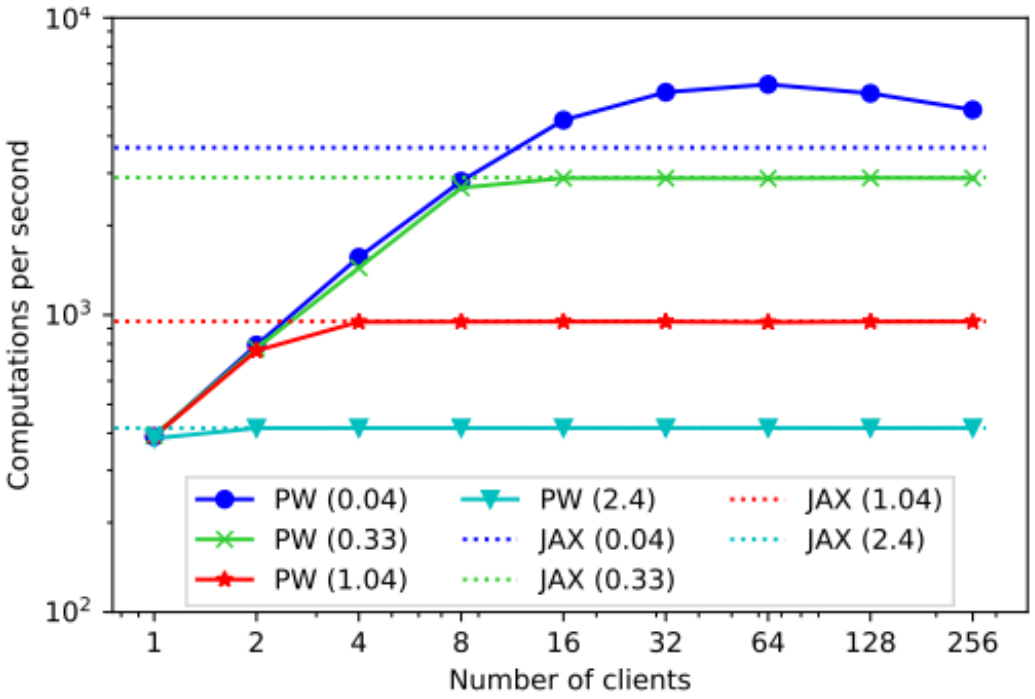
- Why not Jax?
- Multi-controller architectures are fit for one TPU pod.
- Can we run unmodified JAX code?
- Tracer @pw.program

```
def get_devices(n):
  """Allocates `n` virtual TPU devices on an island."""
  device_set = pw.make_virtual_device_set()
  return device_set.add_slice(tpu_devices=n).tpus

a = jax.pmap(lambda x: x * 2., devices=get_devices(2))
b = jax.pmap(lambda x: x + 1., devices=get_devices(2))
c = jax.pmap(lambda x: x / 2., devices=get_devices(2))

@pw.program  # Program tracing (optional)
def f(v):
  x = a(v)
  y = b(x)
  z = a(c(x))
  return (y, z)

print(f(numpy.array([1., 2.])))
# output: (array([3., 5.]), array([2., 4.]))
```

Evaluation


Computations per second
10^4
10^3
10^2
10^1
10^0
JAX-F
JAX-O
PW-O
Ray-O
PW-F
Ray-F
Ray-C
TF-O
PW-C
TF-C
2
4
8
16
32
64
128
256
512
Number of hosts

Evaluation


10^4
10^3
10^2
Computations per second
1
2
4
8
16
32
64
128
256
Number of clients
PW (0.04)
PW (0.33)
PW (1.04)
PW (2.4)
JAX (0.04)
JAX (0.33)
JAX (1.04)
JAX (2.4)

Review

- Strengths and Impact
  - Aims to introduce a new era of distributed ML frameworks that revert to the single-controller architecture
  - Efficient attempt to coordinate multiple islands of TPUs
- Areas of improvement
  - The Resource Manager establishes a 1-to-1 mapping between an accelerator and its virtual counterpart.
  - Data-driven control flow is not handled using asynchronous calls
  - Functions that do not comply with regular compiled/XLA suffer from control-plane latency
  - Unexplored avenue for optimizing accelerator mapping

PaLM Study Case

- 540 billion parameters
- 6144 TPU v4 chips
- 2 months of training with PATHWAYS
- Results learnt on hundreds of language semantics benchmarks
- Outperforms average human on BIG-bench benchmark

Questions?

References

- Barham, Paul, et al. "Pathways: Asynchronous distributed dataflow for ML." Proceedings of Machine Learning and Systems 4 (2022): 430-449.
- MLSys22: https://mlsys.org/virtual/2022/oral/2146
- Clarke, Lyndon, Ian Glendinning, and Rolf Hempel. "The MPI message passing interface standard." Programming environments for massively parallel distributed systems. Birkhäuser, Basel, 1994. 213-218.
- Chowdhery, Aakanksha, et al. "Palm: Scaling language modeling with pathways." arXiv preprint arXiv:2204.02311 (2022).