KiSTi 한국과학기술정보연구원 Korea Institute of Science and Technology Information www.kisti.re.kr

UST

NLP Seminar

Pathways

2022. 06. 03

KISTI - UST IKJE CHOI

Pathways

- Title : Pathways: Asynchronous Distributed Dataflow for ML
- Google Scholar

Pathways: Asynchronous distributed dataflow for ML

P Barham, A Chowdhery, J Dean… - Proceedings of …, 2022 - proceedings.mlsys.org

We present the design of a new large scale orchestration layer for accelerators. Our system, Pathways, is explicitly designed to enable exploration of new systems and ML research ideas, while retaining state of the art performance for current models. Pathways uses a sharded dataflow graph of asynchronous operators that consume and produce futures, and efficiently gang-schedules heterogeneous parallel computations on thousands of accelerators while coordinating data transfers over their dedicated interconnects. Pathways …

☆ Save Cite Cited by 6 Related articles All 3 versions »

Pathways

- Pathways
  - ✓ A new system built for distributed ML.
  - ✓ PATHWAYS uses a client-server architecture that enables PATHWAYS's runtime to execute programs on system-managed islands of compute on behalf of many clients.
  - ✓ PATHWAYS's programming model makes it easy to express non-SPMD computations and enables centralized resource management and virtualization to improve accelerator utilization.
  - ✓ SPMD –
    single program, multiple data (SPMD) is a technique employed to achieve parallelism. Tasks are split up and run simultaneously on multiple processors with different input in order to obtain results faster.

The limitations of current distributed ML systems

- Limitations
  - ✓ Distributed ML systems for training state-of-the-art SPMD models often adopt a multi-controller architecture where the same client executable is run directly on all the hosts in the system, taking exclusive ownership of the resources on those hosts for the duration of the program execution.

Pathways Programming Model

```
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.]))
```

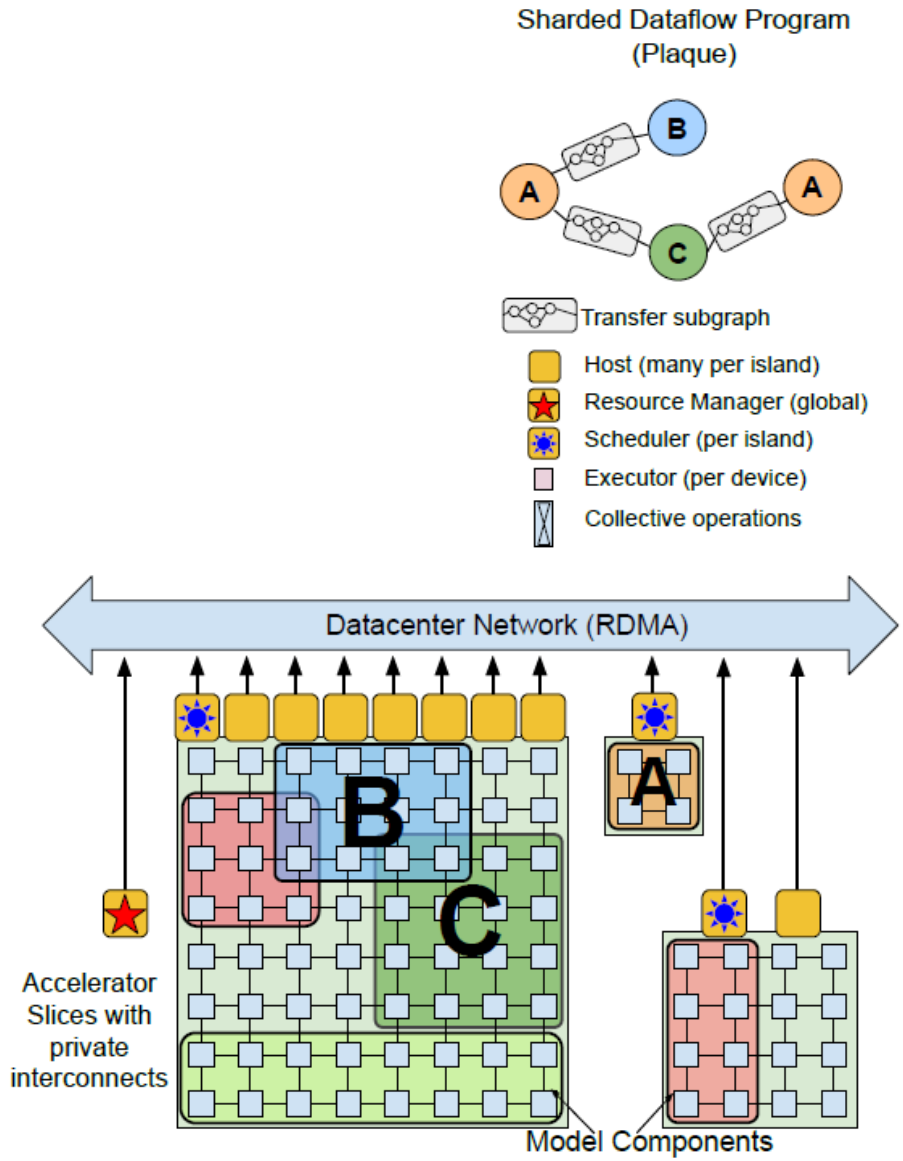
Pathways Programming Model

- virtual devices
  - ✓ A PATHWAYS user may request sets of “virtual devices”, with optional constraints on the device types, locations or interconnect topology, and is then able to place specific compiled functions on those devices

- program tracer
  - ✓ also implemented a new program tracer (Figure 2) that a user can wrap around a block of Python code that calls many compiled functions. The tracer generates a single PATHWAYS program where each compiled function is represented by a computation node in a dataflow graph.

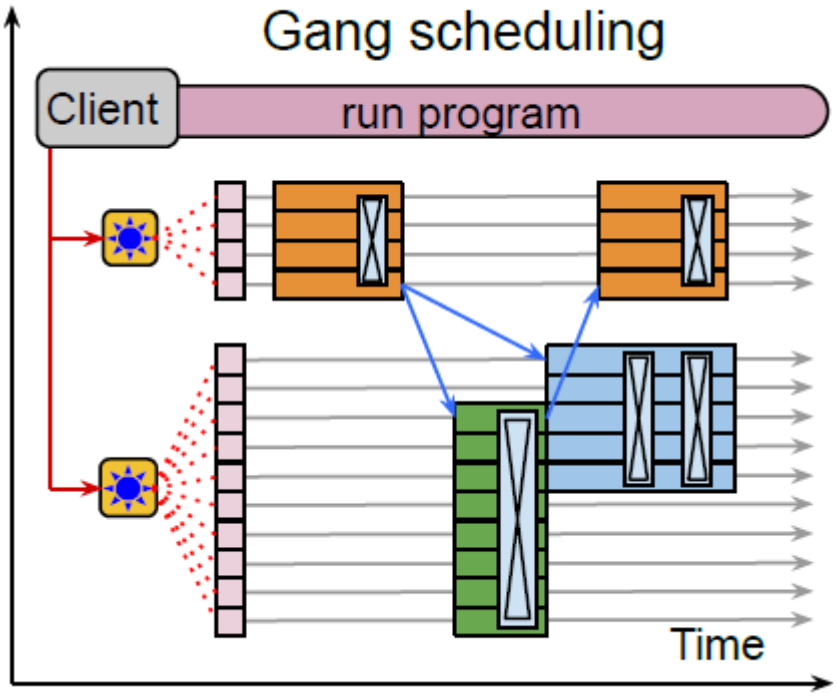
Pathways System Architecture

- Resource Manager
  - ✓ A PATHWAYS backend consists of a set of accelerators grouped into tightly-coupled islands that are in turn connected to each other over DCN (Figure 3). PATHWAYS has a “resource manager” which is responsible for the centralized management of devices across all of the islands.



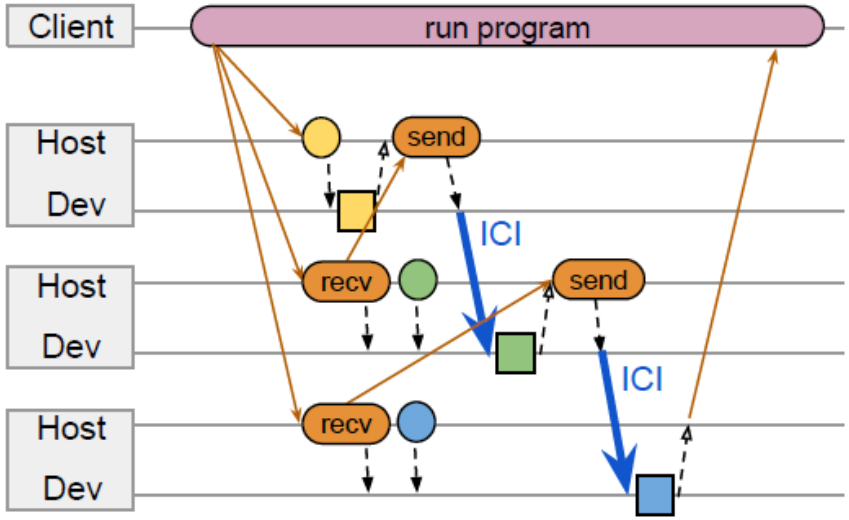
Pathways System Architecture

- Gang Scheduling
  - ✓ The PATHWAYS runtime includes a centralized scheduler per island that consistently orders all of the computations in the island.



Pathways System Architecture

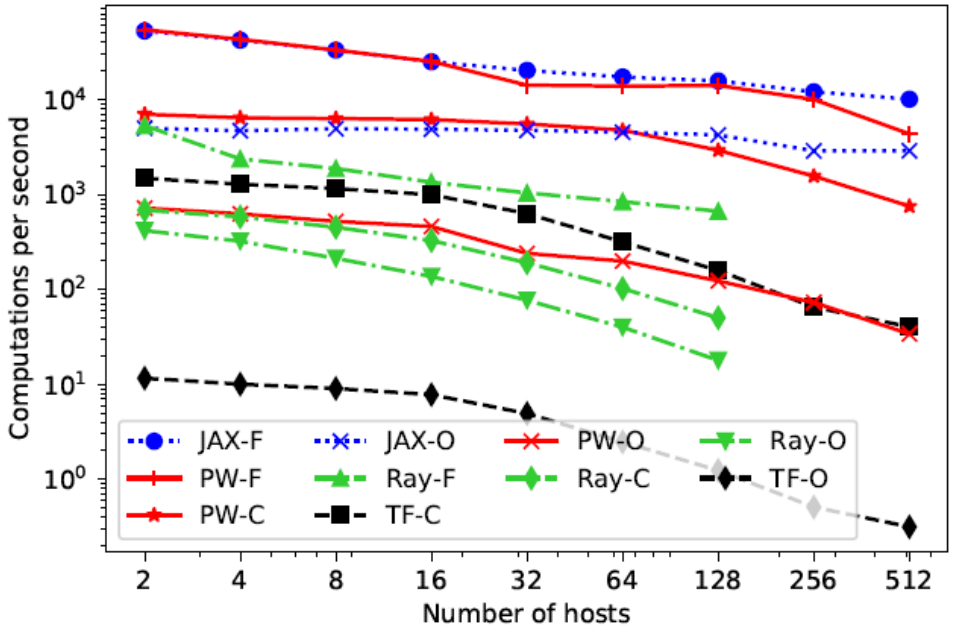
- Parallel asynchronous Dispatch
  - ✓ PATHWAYS treats parallel scheduling as an optimization and falls back to the traditional model when a node's resource requirements are not known until a predecessor computation has completed



(b) Parallel dispatch

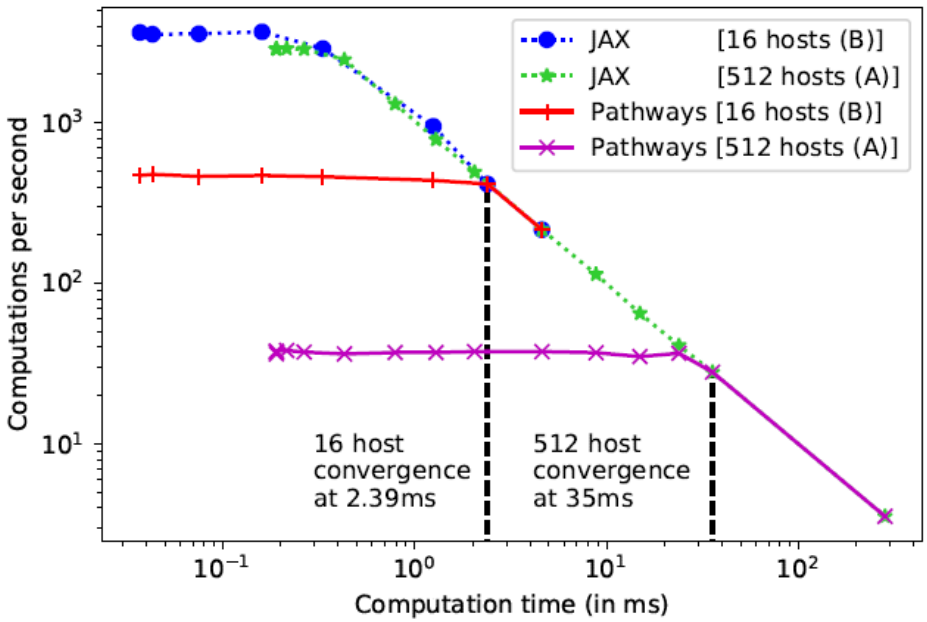
Evaluation

- ✓ Dispatch overhead of PATHWAYS compared to TF, JAX, and Ray. PATHWAYS outperforms single-controller systems like TF and Ray on all configurations, and matches the performance of multi-controller JAX in Fused (-F) and Chained (-C) configurations for up to 1000 and 256 TPU cores, respectively. Each computation comprises a single scalar All Reduce followed by a scalar addition.



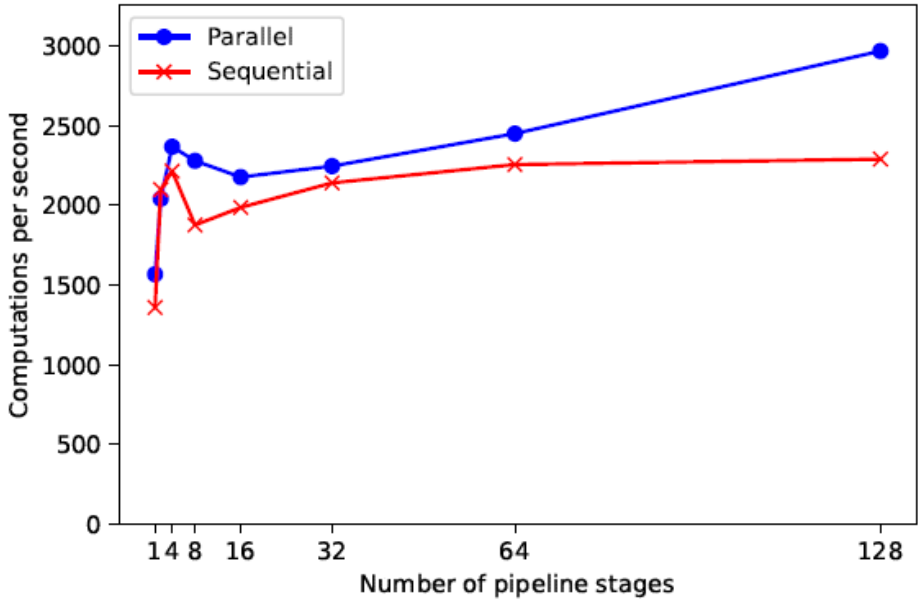
Evaluation

- ✓ Smallest computation to match throughput between PATHWAYS and JAX, masking the single-controller overhead. PATHWAYS matches JAX throughput for a computation size of at least 2.3 ms for 16 hosts with 128 TPUs on configuration (B), and for a computation size of at least 35 ms for 512 hosts with 2048 TPUs on configuration (A).



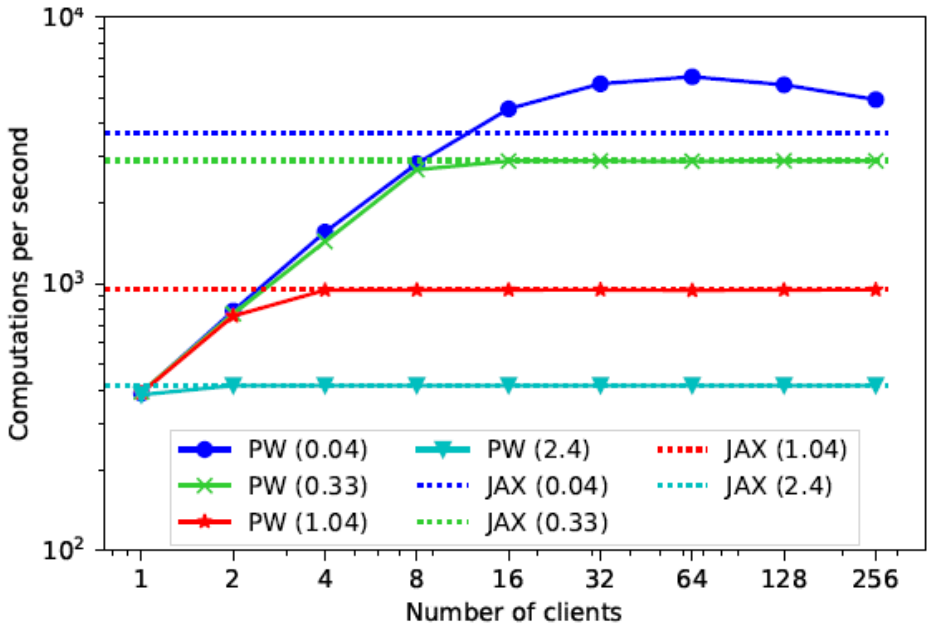
Evaluation

- ✓ Parallel vs. Sequential Async Dispatch in PATHWAYS. Each pipeline stage runs on a different set of 4 TPU cores (different host) transferring data to next stage via ICI. With parallel async dispatch, PATHWAYS amortizes the fixed client overhead and the scheduling overhead for large number of pipeline stages.



Evaluation

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




THANK
YOU