# XProf: An Open, Scalable, and Extensible Profiling System for the Modern ML Stack



May 21st, 2026

Google Cloud

# Exec Summary

## Challenge

Lightweight profiling of large scale ML workloads at varying detail

## Solution

XProf: Google's open-source, scalable, extensible profiler

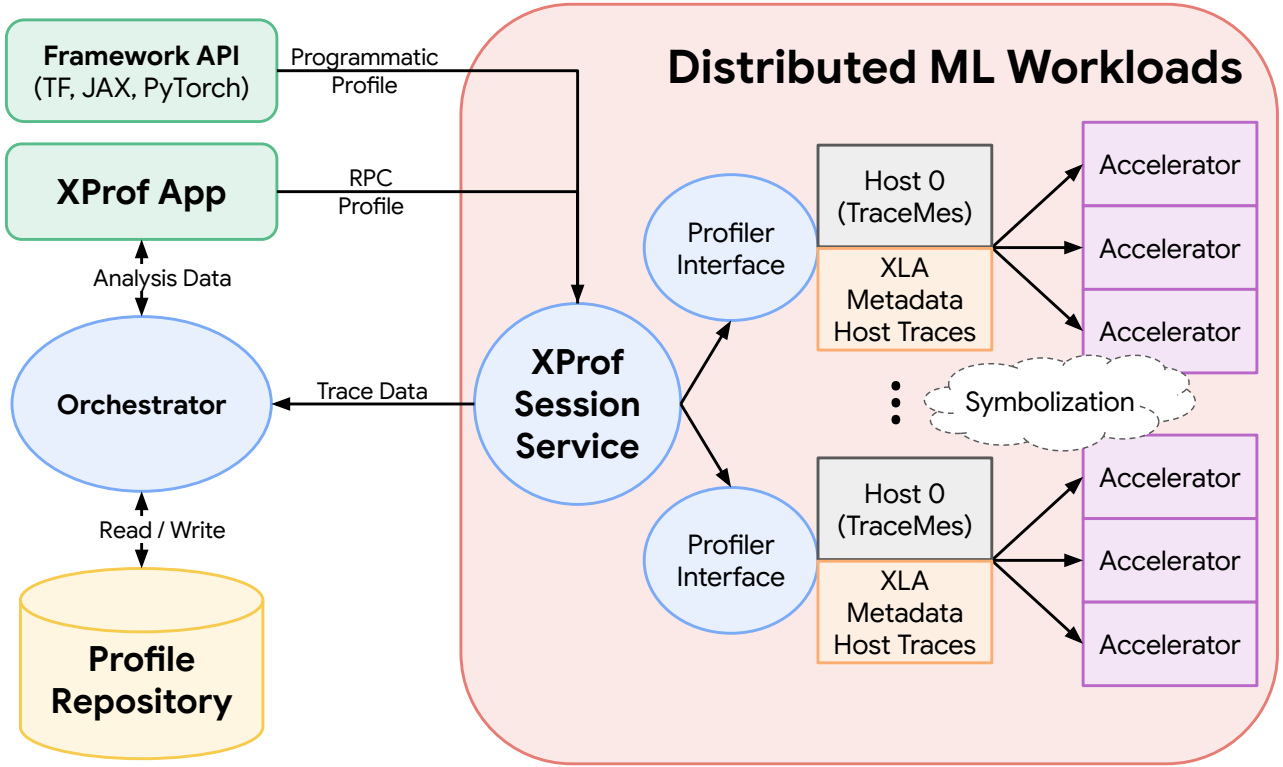
## Innovations

Low trace bandwidth, scalable synchronization, efficient processing

# The Profiling Challenge

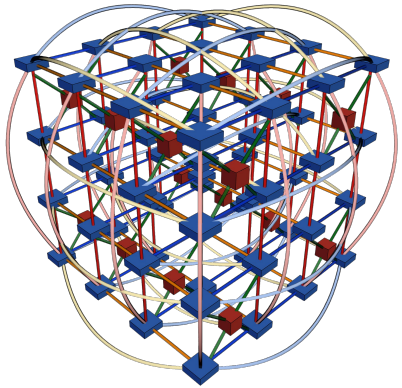
- **Scalability**
  - Fleet / distributed workloads
  - Massive data
- **Discoverability**: Complex ML stack, Host-Device
- **Explainability**: Bridging the abstraction gap across the stack.
- **Adaptability**: Hardware & software co-evolution
- **Fidelity:** The Observer Effect

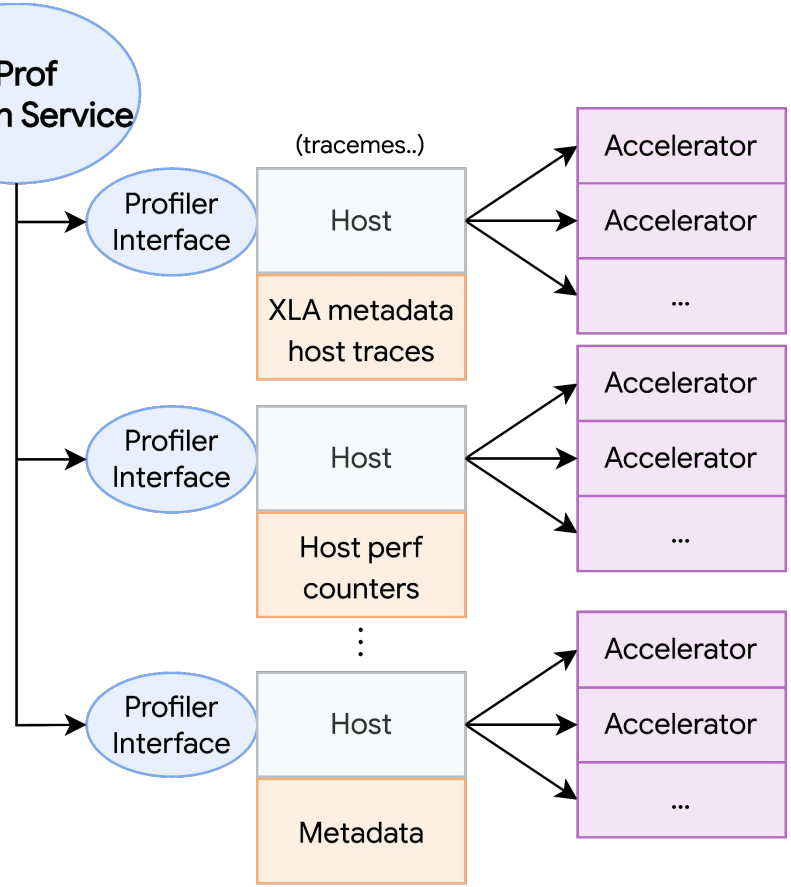
# Introducing XProf



# XProf: Collection

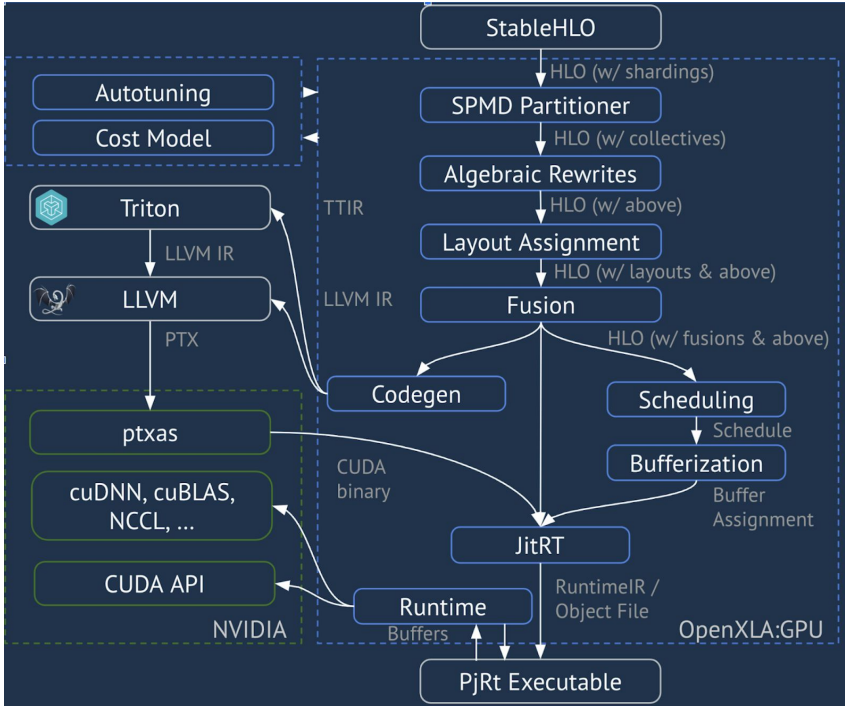
- Lightweight profiler interface
- Distributed collection across hosts
- Collection data
  - XLA compiler metadata
  - Hardware traces
  - Perf counters
  - TraceMe





# XProf: Symbolization

- Maps program counters back to XLA operations
- Decipher complex compiler optimizations
- Insight into various compilation stages
- Basis for high level tooling



# XProf: Analysis

- Multi-level tooling
- Distributed trace processing (MapReduce)
- Correlate high level abstractions with hardware events


**Figure 1:** Screenshot of XProf's Trace Viewer tool

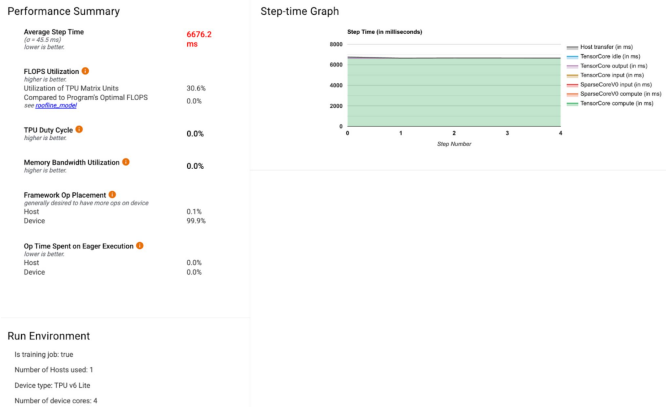



**Figure 2:** Screenshot of XProf's Overview Page

# Multi-Tiered Visibility

| Analysis Level | Target Focus | Host (CPU) Tools | Device (Accelerator) Tools |
|---|---|---|---|
| High-Level | Model Performance | Input Pipeline, High Level Op Tracing | Roofline Model |
| Intermediate | System Interaction | Host TraceMes | Trace Viewer, HLO Op Profile, Memory Viewer, Graph Viewer |
| Low-Level | Hardware State | Trace Viewer, Low Level Op Tracing, Host Perf Counters | Perf Counters, Utilization Viewer |

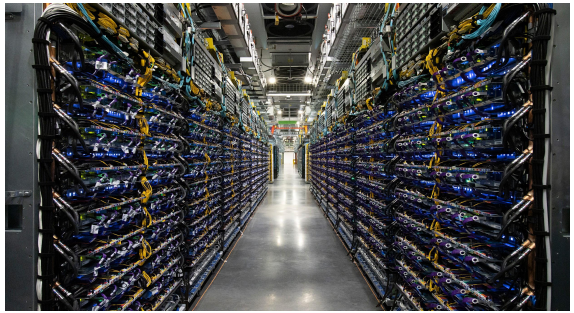
# Core Innovations

## Lightweight Tracing

**TraceMe** instrumentation defers processing to maintain kilobyte-scale tracing.

## Precision Clocking

**Global Timestamp Counter (GTC)** to synchronize thousands of accelerators.



## Scalable Architecture

**MapReduce** backend orchestrate multiple hosts in large scale profiles.

# TraceMe: Ultra Low Overhead

## Surgical Precision

Developer defined regions of interest

## Deferred Correlation

“Connect the dots later” approach



## Non-Blocking Design

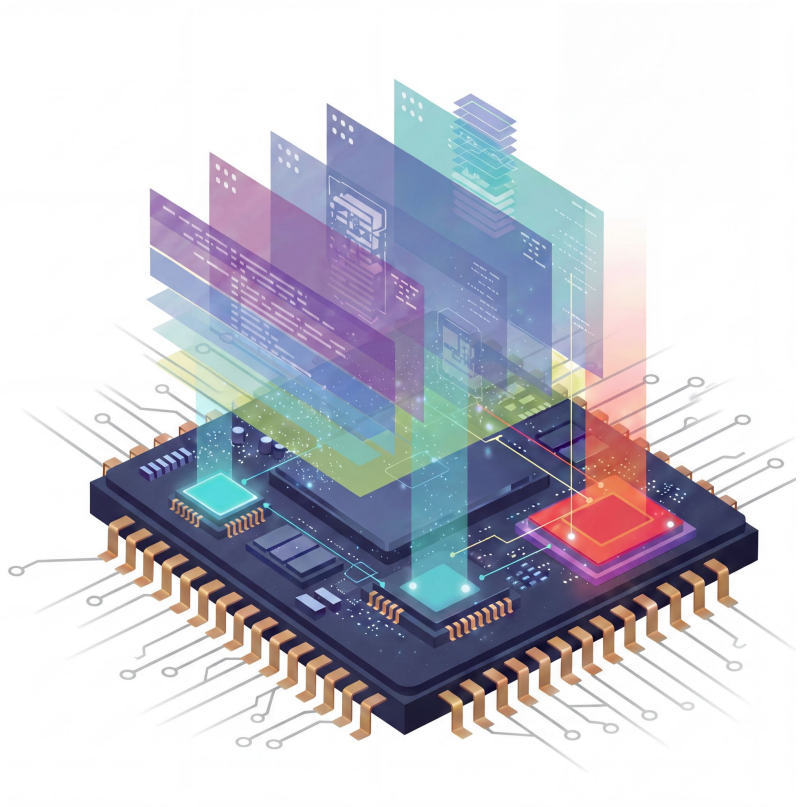
Lock-free, atomic, thread-local

## Low Trace Volume

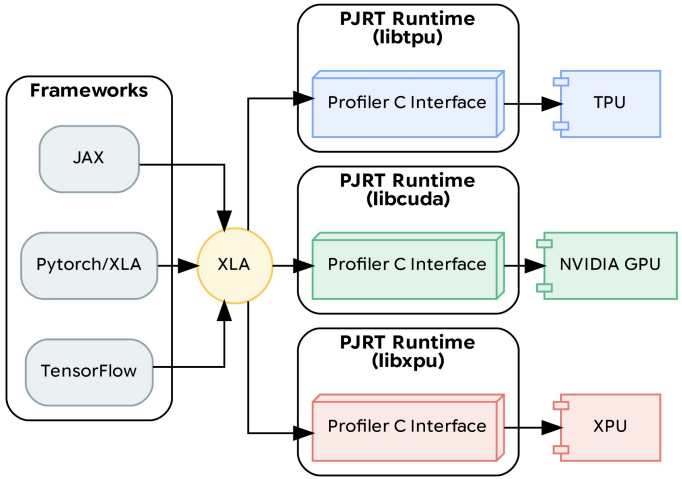
Tracing on the order of kilobytes

# Deep Compiler Integration

- Seamless integration with ML compiler (XLA)
- Correlates and bridging the gap between abstractions:
  - Framework / Python source code
  - MLIR
    - High level ML operations
    - Low level ML operations
  - Hardware runtime counters



# Open & Extensible Framework




### Device Agnostic Design

- Leverages OpenXLA PJRT
- Decouple: profiler - specific hardware accelerators.
- Framework profiling support (JAX, PyTorch) without rigid hardware dependencies.

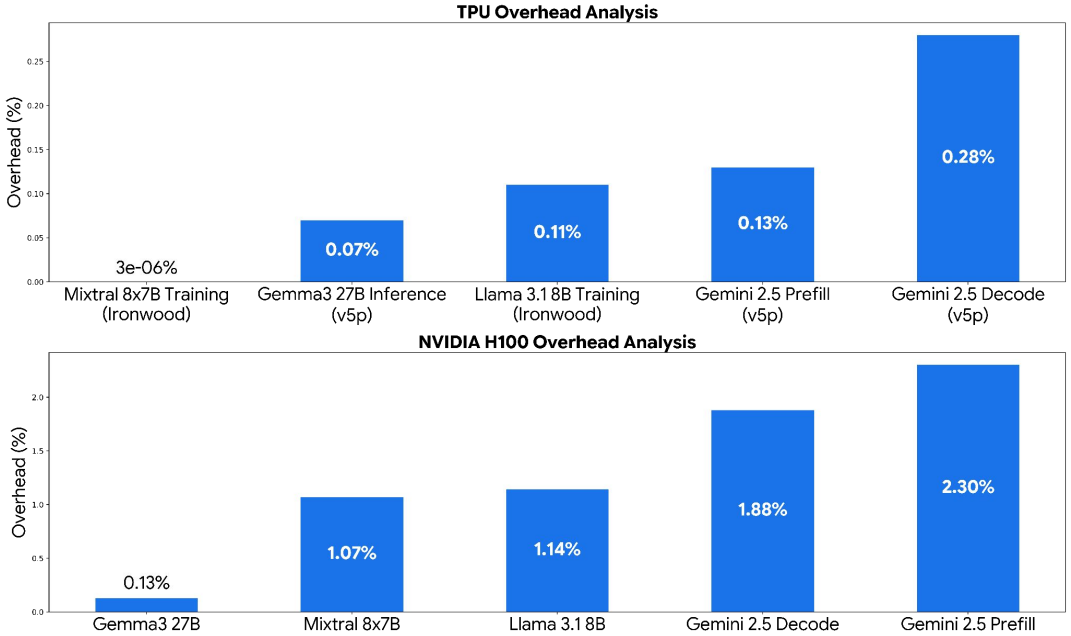
### Unified Interface

- C Profiler API for third-party hardware vendors.
- Support for TPUs, GPUs (NVIDIA, AMD), and custom AI chips supporting XLA.
- Accelerator Examples:
  - Amazon Trainium
  - Intel GPUs
  - AMD GPUs

# Host Collection Overhead

## LOW OVERHEAD ON SOTA MODELS

- Production safe
- Consistently maintaining well **< 0.3%** overhead on TPUs
- Highly efficient profiles on GPUs



# Case Study: Power & Thermal

**Smoothing Peak Demands**

- Compute-intensive and communication intensive phases
- Trace viewer correlation with power metrics
- Complex hardware interactions

**10°C**

**Reduction in Thermal Swings**

Join our team! Build the world's best ML infrastructure.

https://g.co/jobs/xprof

# https://github.com/openxla/xprof


Charles Alaras


Yin Zhang


Jiya Zhang


Clive Verghese

# Thank you

Google Cloud