PyTorch CONFERENCE 2025

TraceLens: Democratizing AI Performance Analysis

- TraceLens transforms raw trace data from PyTorch (and JAX) into actionable insights—no code changes, no re-runs.
- It bridges the gap between profilers that dump gigabytes of raw data and the engineers who need to understand where time is spent in the workload and how efficiently the hardware is used. TraceLens lets you analyze performance from high-level model blocks down to individual GPU kernels.
- Built as an analysis layer, not an instrumentation tool, TraceLens operates on existing traces to expose inefficiencies in compute, memory, and communication.
- Today, 100+ AMD engineers and external partners use TraceLens to triage regressions, design performance-aware architectures, and share precise performance data securely—democratizing performance analysis for modern AI workloads.

https://github.com/AMD-AGI/TraceLens

Contact: Adeem Jassani
Email: ajassani@amd.com

Trace2Tree: A powerful IR for Deeper Analysis

- Raw GPU traces reveal what ran but not why. Kernel names alone cannot explain which model operator launched them, their data shapes, or whether they belong to forward or backward computation.
- Trace2Tree bridges this gap by converting framework traces from PyTorch (and JAX) into a hierarchical tree representation that links every GPU kernel back to its high-level nn.Module, operator, and runtime dispatch.
- This tree is a powerful intermediate representation (IR) that captures both structural and performance semantics of AI workloads. It forms the foundation of many TraceLens features.

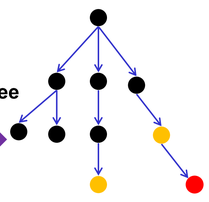
PyTorch python frontend/ C++ backend

Hip/Cuda runtime

GPU kernel

PyTorch profiler ".json" trace file

Trace2Tree



This Tree is a powerful IR for further analysis

NN Module Tree

```
.....................................................(GPU Time µs)
└── nn.Module: DeepseekV2DecoderLayer_2 ..... (10233.51)
    ├── nn.Module: RMSNorm_0 ..... (148.46)
    ├── nn.Module: DeepseekV2AttentionMLA_2 ..... (6775.96)
    │   ├── nn.Module: ReplicatedLinear_4 ..... (817.78)
    │   ├── nn.Module: RMSNorm_9 ..... (19.67)
    │   ├── nn.Module: ColumnParallelLinear_2 ..... (268.80)
    │   ├── nn.Module: ReplicatedLinear_5 ..... (469.98)
    │   ├── nn.Module: RMSNorm_10 ..... (10.21)
    │   ├── nn.Module: DeepseekScalingRotaryEmbedding_0 ..... (139.27)
    │   ├── nn.Module: RadixAttention_2 ..... (3204.62)
    │   ├── nn.Module: RowParallelLinear_4 ..... (1446.40)
    │   └── Non-nn.Module GPU Ops ..... (399.22)
    ├── nn.Module: RMSNorm_11 ..... (154.43)
    └── nn.Module: DeepseekV2MLP_2 ..... (3154.66)
        ├── nn.Module: MergedColumnParallelLinear_2 ..... (1592.94)
        ├── nn.Module: SiluAndMul_2 ..... (84.44)
        └── nn.Module: RowParallelLinear_5 ..... (1477.28)
```

Example Trace Tree

```
└── (python) nn.Module: Linear_75
    └── (python) forward @ linear.py:124
        └── (cpu) aten::linear
            ├── (cpu) aten::to
            │   └── (cpu) aten::_to_copy
            │       └── (cpu) aten::copy_
            │           └── (cuda_runtime) hipLaunchKernel
            │               └── (kernel) elementwise_kernel...
            ├── (cpu) aten::to
            │   └── (cpu) aten::_to_copy
            │       └── (cpu) aten::copy_
            │           └── (cuda_runtime) hipLaunchKernel
            │               └── (kernel) elementwise_kernel...
            └── (cpu) aten::linear
                └── (cpu) aten::addmm
                    ├── (cuda_runtime) hipLaunchKernel
                    │   └── (kernel) elementwise_kernel...
                    └── (cuda_runtime) hipExtModuleLaunchKernel
                        └── (kernel) Cijk_Alik_Bljk_BBS_BH_Bias_HAS_SAV...
```

Morphological Trace Diff

```
(cpu) aten::convolution
└── (cpu) aten::_convolution
      |
      |--- Trace 1 (e.g., CUDA Backend) -----------
      |
      - (cpu) aten::cudnn_convolution
      |   └── (runtime) cuLaunchKernelEx
      |       └── (kernel) nvjet_tst_...
      - (cpu) aten::add_
      |   └── (runtime) cudaLaunchKernel
      |       └── (kernel) elementwise_kernel...
      |
      |--- Trace 2 (e.g., ROCm Backend) -----------
      |
      + (cpu) aten::miopen_convolution
      |   └── (runtime) hipExtModuleLaunchKernel
      |       └── (kernel) igemm_fwd_gtcx3_...
      + (cpu) aten::add_
      |   └── (runtime) hipLaunchKernel
      |       └── (kernel) elementwise_kernel...
```

Hierarchical Performance Breakdowns: Pinpoint Root Cause

TraceLens organizes performance data in a top-down hierarchy, letting you drill down from the overall GPU timeline to specific operations and input shapes.
•**GPU Timeline Summary:** Gives a high-level breakdown of total GPU time across computation, communication, and idle phases—revealing if your workload is compute-bound, communication-bound, or CPU-bound.
•**Op Category Summary:** Aggregates kernel time by operation family (e.g., GEMM, SDPA, elementwise) to quickly identify which types of operations dominate runtime.
•**Op Name Summary:** Lists the most time-consuming operations, ranking them by total kernel time to pinpoint hotspots.
•**Op Arg Summary:** Breaks each operator by input shapes and dtypes, exposing shape-specific bottlenecks and regression sources.

GPU timeline summary

type	time ms	percent
computation_time	56305.19	99.30
exposed_comm_time	240.88	0.42
exposed_memcpy_time	14.44	0.03
busy_time	56560.52	99.75
idle_time	143.16	0.25
total_time	56703.68	100.00
total_comm_time	17203.43	30.34
total_memcpy_time	14.47	0.03

Op category summary

op category	Count	Kernel time (ms)	Percent (%)
GEMM	3046	45706.79	79.47
SDPA_fwd	320	4011.13	6.97
SDPA_bwd	160	3282.01	5.71
triton	4008	1938.21	3.37
multi_tensor_apply	644	1278.03	2.22
other	1000	839.26	1.46
elementwise	2104	362.71	0.63
reduce	326	99.61	0.17

Op Name summary

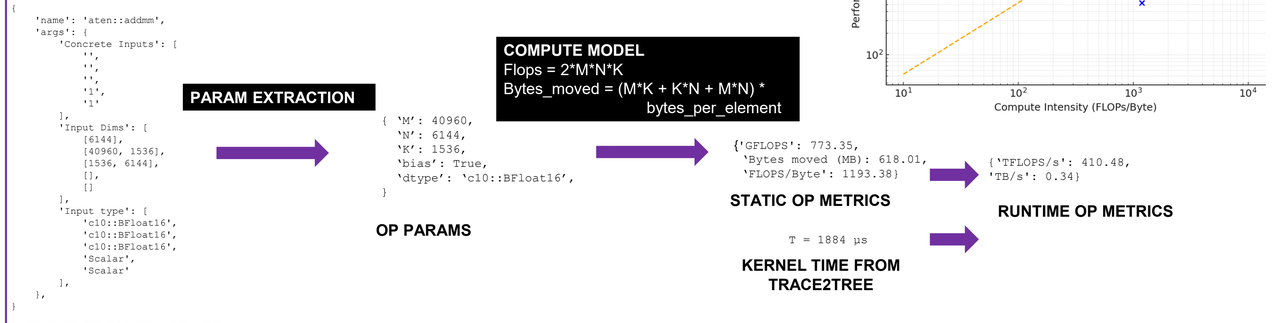
Op name	Count	Kernel time (ms)	Percent (%)	Cum Percent (%)
aten::mm	3046	45706.79	79.47	79.47
flash_attn::_flash_attn_forward	320	4011.13	6.97	86.44
flash_attn::_flash_attn_backward	160	3282.01	5.71	92.15
aten::_foreach_copy_	640	1203.75	2.09	94.24
triton_poi_fused_add_fill_mul_sigmoid_silu_sub_7	160	458.78	0.80	95.04
aten::_chunk_cat	162	401.01	0.70	95.73
aten::split_with_sizes_copy	320	365.63	0.64	96.37
aten::mul	962	276.39	0.48	96.85
triton_poi_fused_mul_silu_7	160	270.23	0.47	97.32
triton_red_fused__to_copy_add_div_mean_mul_pow_rsqrt_sum_8	160	164.36	0.29	97.60

Op arg summary

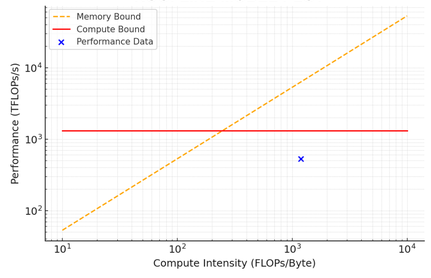
name	Input Dims	Input type	Input Strides	count	Kernel time mean(ms)
aten::mm	((24576, 8192), (8192, 28672), (24576, 28672))	('c10::BFloat16', 'c10::BFloat16', 'c10::BFloat16')	((8192, 1), (1, 8192), (28672, 1))	640	22255.71
aten::mm	((28672, 24576), (24576, 8192), (28672, 8192))	('c10::BFloat16', 'c10::BFloat16', 'c10::BFloat16')	((1, 28672), (8192, 1), (8192, 1))	320	19729.18
aten::mm	((24576, 28672), (28672, 8192), (24576, 8192))	('c10::BFloat16', 'c10::BFloat16', 'c10::BFloat16')	((28672, 1), (8192, 1), (8192, 1))	320	18973.62
aten::mm	((24576, 8192), (8192, 28672), (24576, 28672))	('c10::BFloat16', 'c10::BFloat16', 'c10::BFloat16')	((8192, 1), (28672, 1), (28672, 1))	160	21028.56
aten::mm	((8192, 24576), (24576, 28672), (8192, 28672))	('c10::BFloat16', 'c10::BFloat16', 'c10::BFloat16')	((1, 8192), (28672, 1), (28672, 1))	160	20040.82

Compute Modeling: Is the Workload Using the Hardware Efficiently?

- A kernel's duration tells you how long it took, but not how efficiently it used the hardware. To understand efficiency, you need to compare the work performed (FLOPs and Bytes moved) against that duration.
- TraceLens automates this by integrating a compute model for popular deep learning operations.
- For key ops like GEMM, Convolution, and Scaled Dot-Product Attention, TraceLens parses arguments like tensor shapes directly from the trace. It then applies a theoretical model to translate raw timings into efficiency metrics like TFLOP/s and TB/s.

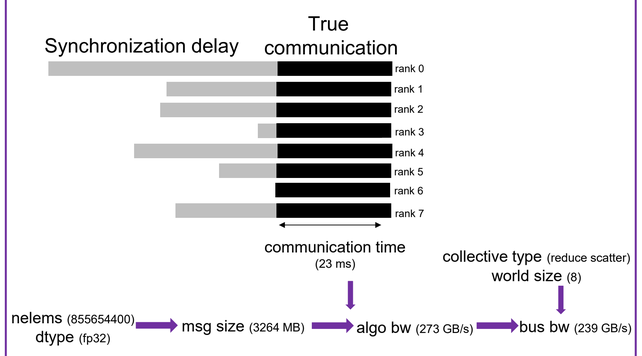


ROOFLINE MODEL PLOT



Multi-GPU Communication Analysis: Finding True Scaling Bottlenecks

- For multi-GPU workloads, a key question is how much of the slowdown comes from the network. Total collective time can be misleading—it often includes periods where faster GPUs wait for slower ones to reach the same point, a phenomenon known as inter-rank synchronization skew.
- TraceLens dissects collective operations to separate this skew from the true communication cost, allowing you to pinpoint scaling inefficiencies and measure the real performance of your interconnect under your specific workload—not just a synthetic benchmark.



Collective name	In msg size (MB)	dtype	comm duration mean (µs)	count	Total latency (ms)	algo bw (GB/s)	bus bw (GB/s)	Sync skew (µs)
allgather	204.00	BFloat16	6041.88	318	1921.32	264.00	231.04	11779.36
reduce_scatter	3264.06	Float	11662.77	160	1866.04	273.43	239.25	60238.77
reduce_scatter	8016.03	Float	22988.50	2	45.98	340.53	297.96	146.48
allgather	501.00	BFloat16	11920.84	2	23.84	41.04	35.91	15405.14
allreduce	~0.00	Float	18.58	6	0.11	~0.00	~0.00	936.63

Event Replay: Isolate and Benchmark Operations

- When you find a slow or problematic operation, the next step is to debug it. This can be difficult, as it often requires the original model, the full data pipeline, and specific inputs just to reproduce the issue. Sharing this complex environment with kernel developers or hardware vendors is often impractical due to IP concerns.
- TraceLens Event Replay feature solves this by generating minimal, self-contained replay scripts directly from trace metadata. It reconstructs the arguments of a target operation—including tensor shapes, data types, and strides—allowing you to reproduce its behavior in isolation.

EVENT FROM TRACE JSON

```
{
    'name': 'aten::addmm',
    'args': {
        'Concrete Inputs': [
            '',
            '',
            '',
            '1',
            '1'
        ],
        'Ev Idx': 75580,
        'External id': 19788,
        'Fwd thread id': 0,
        'Input Dims': [
            [2048],
            [10240, 2048],
            [2048, 2048],
            [],
            []
        ],
        'Input Strides': [
            [1],
            [2048, 1],
            [1, 2048],
            [],
            []
        ],
        'Input type': [
            'c10::BFloat16',
            'c10::BFloat16',
            'c10::BFloat16',
            'Scalar',
            'Scalar'
        ],
        'Record function id': 0,
        'Sequence number': 36606
    },
}
```

REPLAY INFO

```
Operation: aten::addmm
-------------------------------------
Replay IR
  Positional Args:
    • self  :
        Tensor([2048],
               dtype=BFloat16,
               strides=[1])
    • mat1  :
        Tensor([10240, 2048],
               dtype=BFloat16,
               strides=[2048, 1])
    • mat2  :
        Tensor([2048, 2048],
               dtype=BFloat16,
               strides=[1, 2048])

  Keyword Args:
    • beta  : Scalar(1.0)
    • alpha : Scalar(1.0)
-------------------------------------
```

Build Tensors and call function