

CIDS – Center for Interdisciplinary Digital Sciences

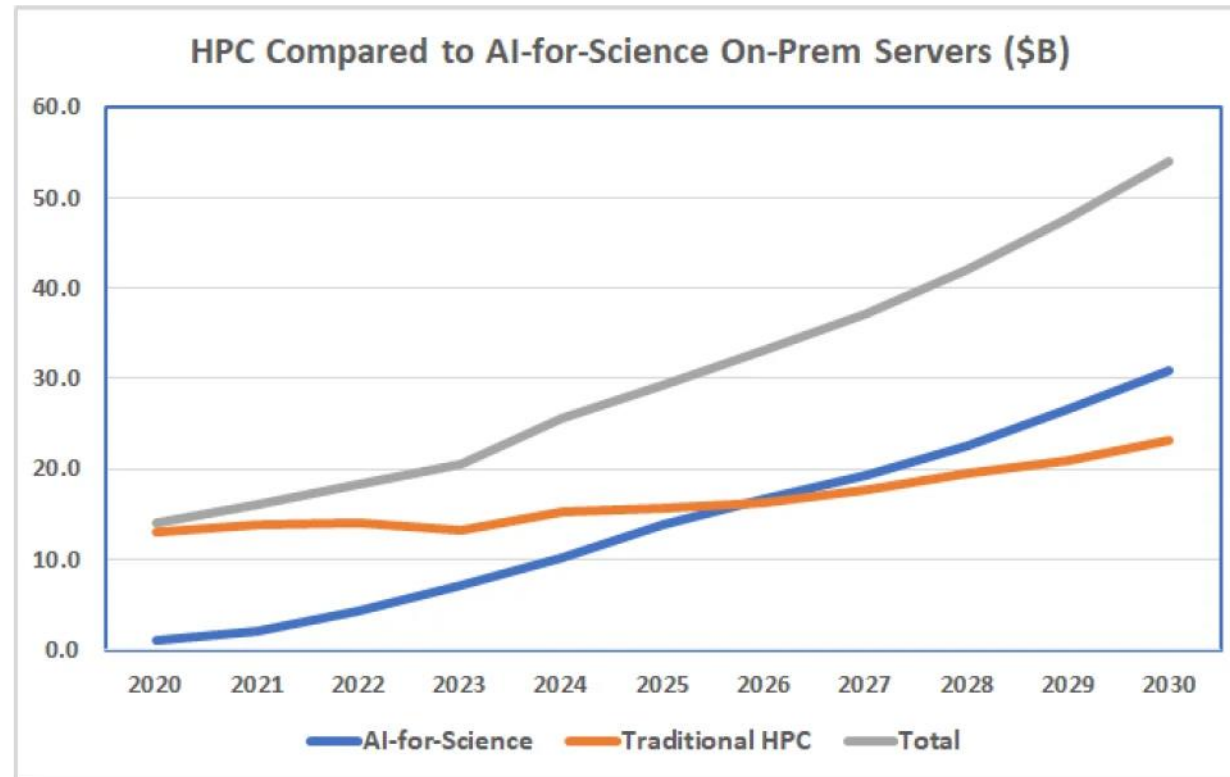
Combining PyTorch Instrumentation APIs with Existing HPC Analysis Tools

Sebastian Döbel, Malte Christian Kuns, Elias Werner

AI - A Growing Force on HPC Systems

Growing demand for AI workloads on HPC infrastructure

Optimize resource utilization to maximize performance and throughput



© Hyperion Research 2026 https://hyperionresearch.com/wp-content/uploads/2026/06/Hyperion-Research_ISC26_Market-Update.pdf

Two Worlds of AI Performance Analysis

Traditional Tools

- Trace Python stack, GPU activity
- System metrics
- Optimized for multi-node scenarios

→ Blackbox at PyTorch level

Built-in Analysis Tools

- Built-in available
- Trace PyTorch operation from compute graph
- Insight into internal states
- Export to CSV or ChromeTrace (json)

→ Blackbox outside PyTorch scope

Related Work

HolisticTraceAnalysis: Analysis of PyTorch performance traces

Meta/Facebook Research. 2025. <https://github.com/facebookresearch/HolisticTraceAnalysis>

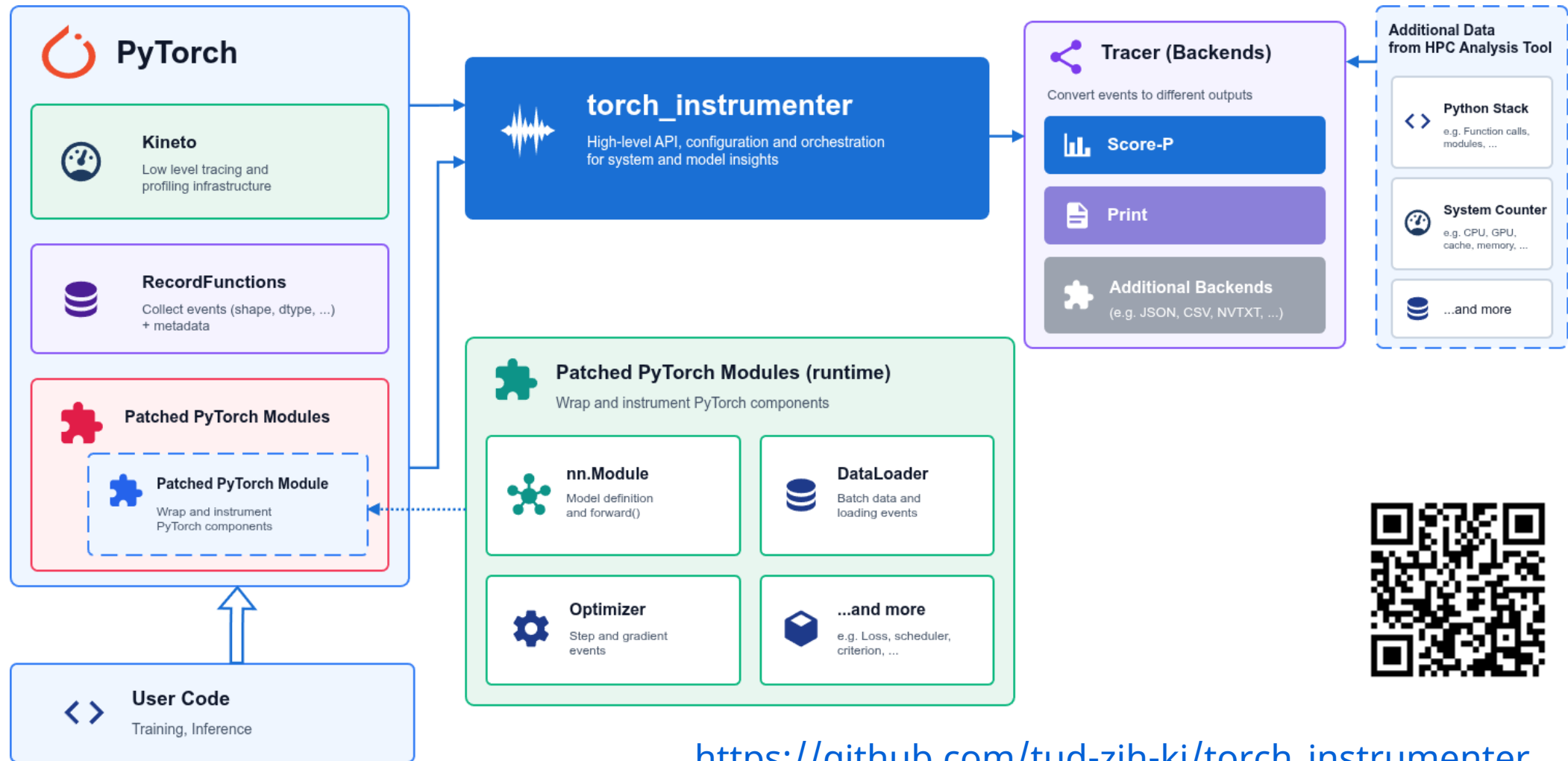
Perseus: Slow down operations outside the critical path

Chung, Jae-Won, et al. "Reducing energy bloat in large model training." Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles. 2024.

Performance Analysis of Jupyter Notebooks

Werner, Elias, et al. "JUmPER: Performance Data Monitoring, Instrumentation and Visualization for Jupyter Notebooks." SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2024.

Architecture



https://github.com/tud-zih-ki/torch_instrumenter

PyTorch RecordFunction-API

Central instrumentation mechanism in PyTorch

- Integrated into PyTorch profiler
- Creates callbacks around operator/function execution
- Hooks into PyTorch's dispatcher/operator execution path
- Allows instrumentation without modifying individual operators

ATen

C++ tensor and operator library

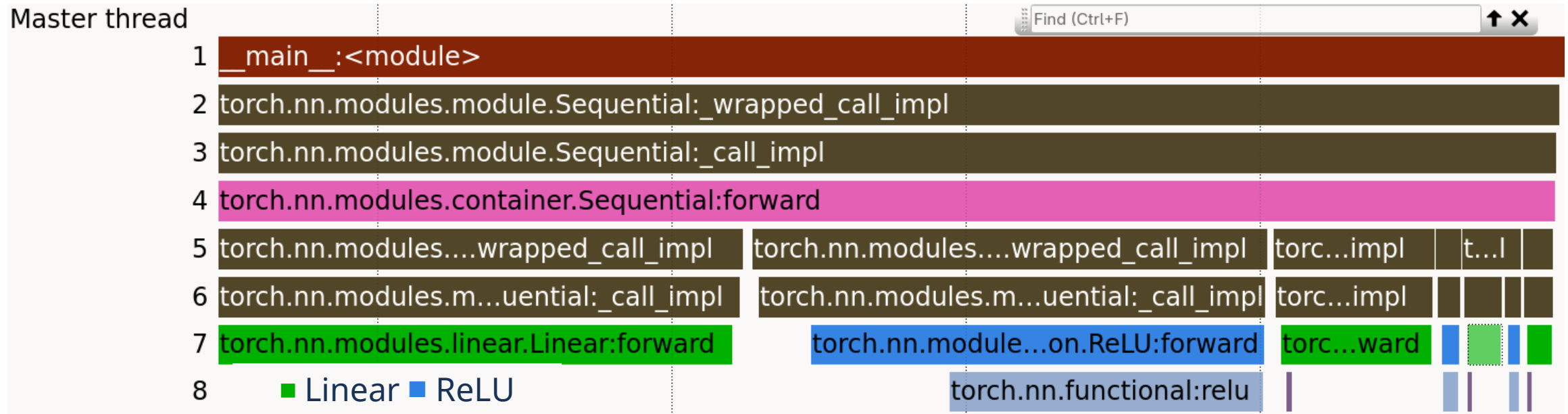
- Most low-level tensor operations are eventually executed as ATen operators
- Next step: backend implementation (CPU kernel, CUDA kernel)

Kineto

PyTorch's profiling backend

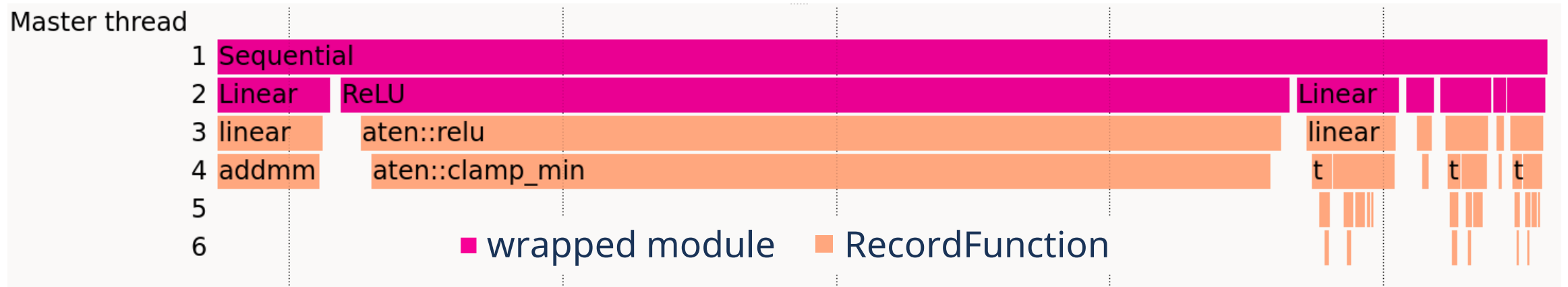
- Recording of GPU activity via CUDA and CUPTI
 - Provides unified CPU $\leftrightarrow$ GPU execution timeline
 - Exposes information beyond the Python layer
-
- Open question if it provides more than Score-P
 - Not yet integrated in torch_instrumenter

Example: Small MLP – Python Stack



Small MLP: Sequential{Linear, ReLU, Linear, ReLU, Linear, ReLU, Linear}

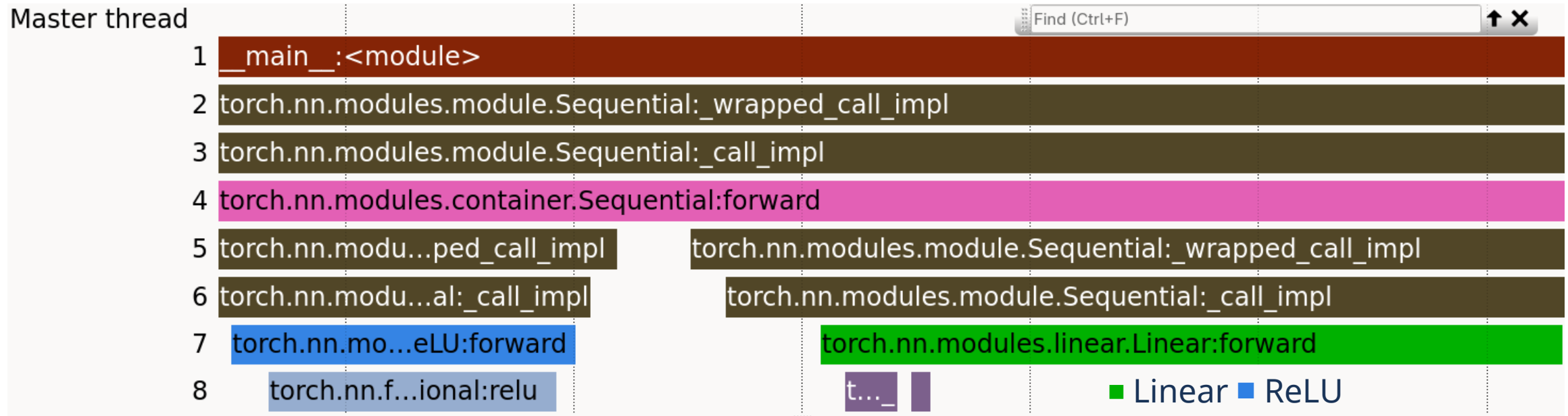
Example: Small MLP – torch_instrumenter



Small MLP: Sequential{Linear, ReLU, Linear, ReLU, Linear, ReLU, Linear}

Example: Small MLP – Python Stack (2)

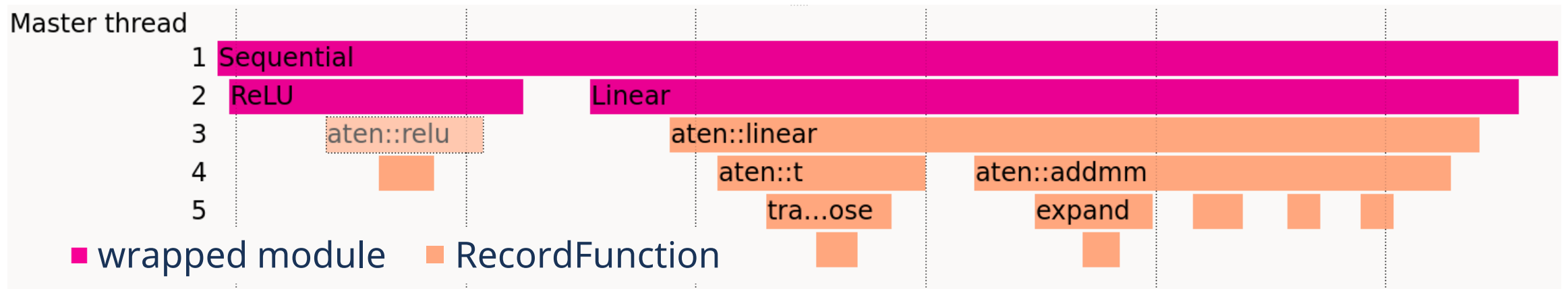
Zoom into the final ReLU-Linear operation sequence



Small MLP: Sequential{Linear, ReLU, Linear, ReLU, Linear, ReLU, Linear}

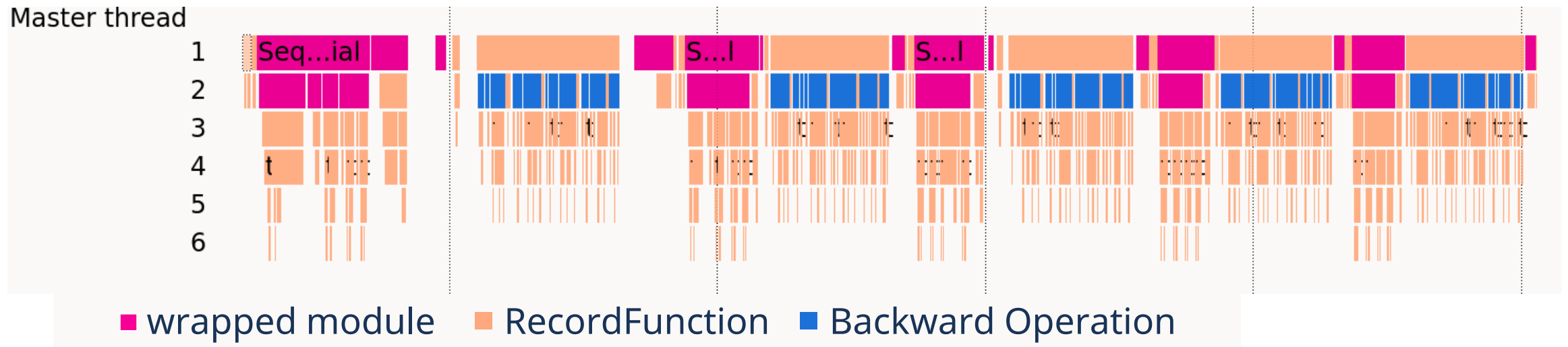
Example: Small MLP – torch_instrumenter (2)

Zoom into the final ReLU-Linear operation sequence



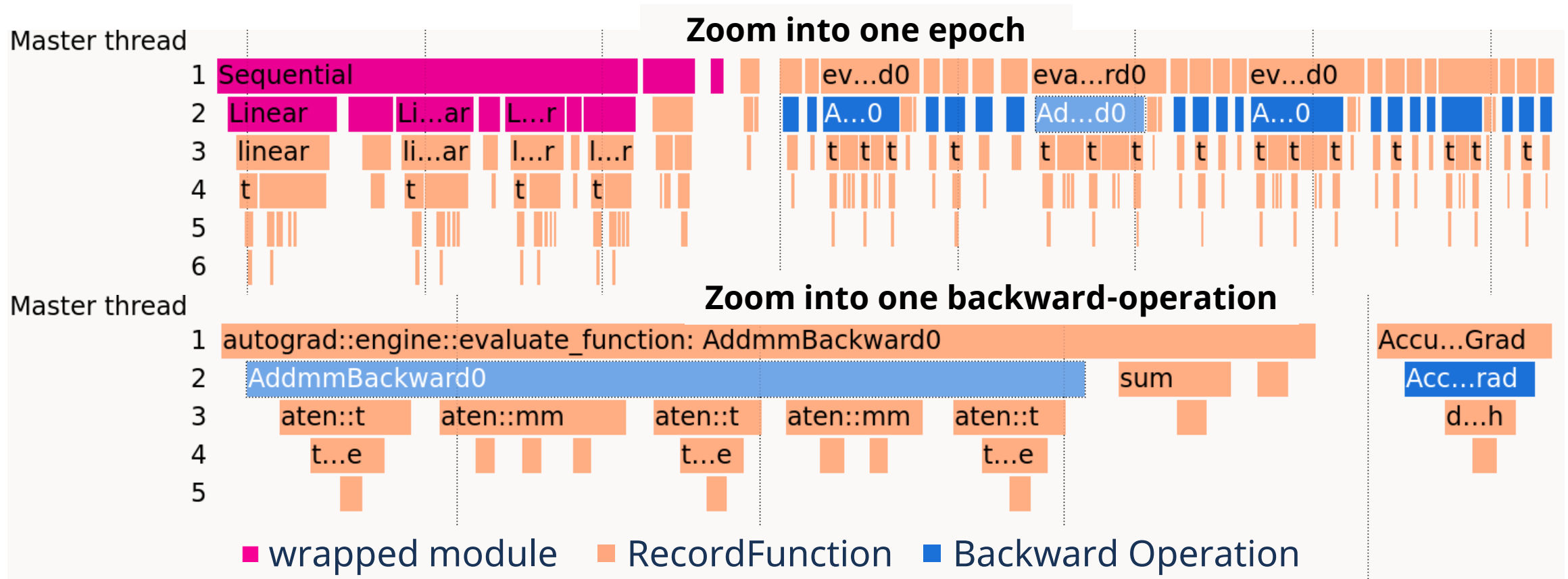
Small MLP: Sequential{Linear, ReLU, Linear, ReLU, Linear, ReLU, Linear}

Example: Training the MLP for 5 Epochs

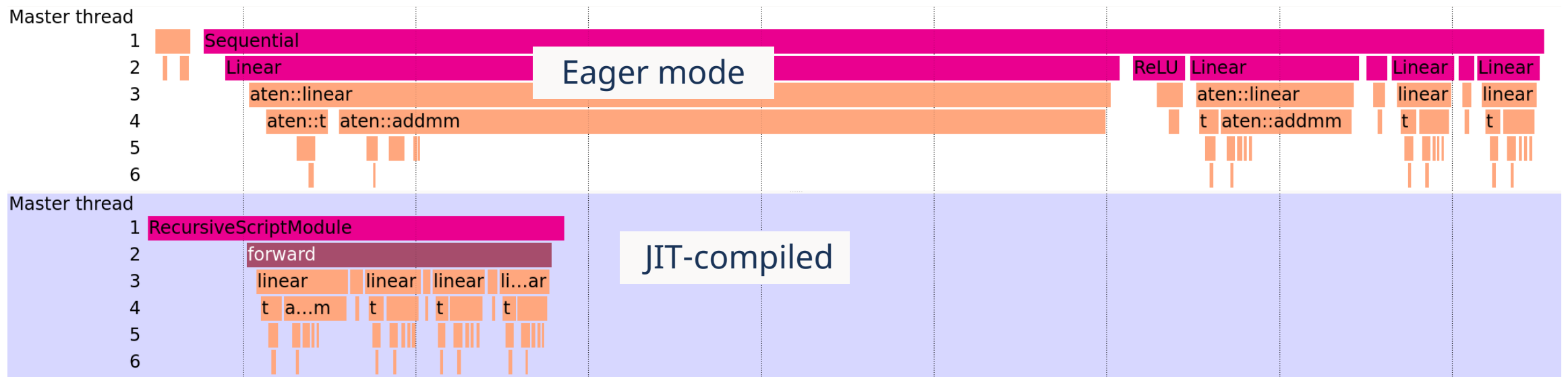


Repeated forward and backward passes across 5 epochs
Iterative update of model parameters to minimize the loss

Example: Training the MLP for 5 Epochs (2)

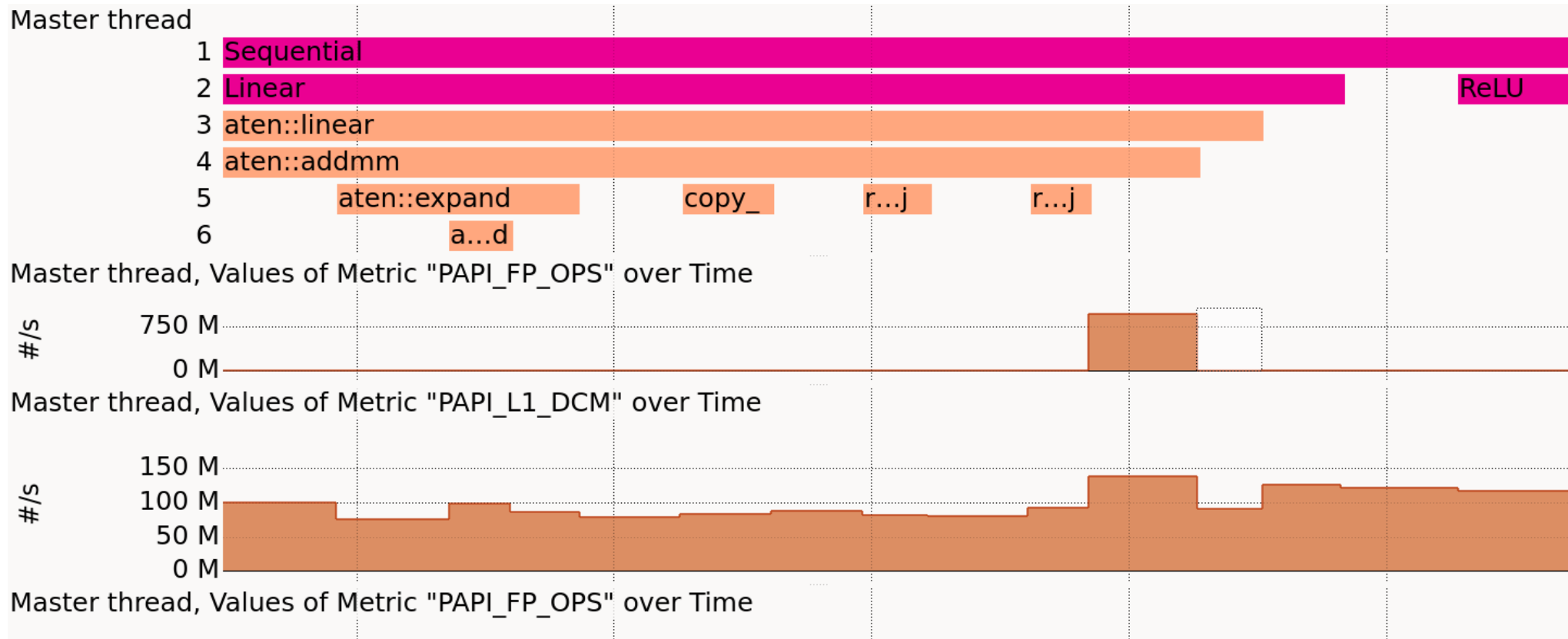


JIT-compilation functions



Supports recording JIT-compiled functions
Initial JIT compilation overhead, followed by significantly faster execution

Correlation with Hardware Counter



Integrate hardware counter measurements into the same trace
Floating-point computation occurs only at the end of *addmm*

Recording Operation Parameters

Process Timeline	
Property	Value
Display	Process Timeline
Type	Function
Location	Master thread
Function	 aten::mm
Function Group	 Pytorch Operation
Interval Begin	0.772496 s
Interval End	0.77257 s
Duration	7.3618e-05 s
Time Range	Set Zoom
Call Level	3
Attributes	
torch.tensors.input	[32,4]:BFloat16;[4,32]:BFloat16
torch.tensors.output	[32,32]:BFloat16

Recording tensor shapes and data types

- Introduces additional overhead
- Extensible to additional parameters e.g. sequence number (Autograd), thread id for async operations, non-tensor operation input
- Still exploring approaches for evaluating function attributes

Summary

Integrate PyTorch operations into existing analysis tools (e.g. Score-P)

Expose C++ ATen tensor operations

Capture operation context: shapes, data types

Minimal source-code modifications — just enable the instrumentation module

Future Work

Still under development

- Kineto Profiler for GPU event recording
- Support for additional trace backends
- There are frameworks beyond PyTorch – Jax, vLLM, ...

Thank you for your attention