

Jinjie Liu / Chunlei Men

One native module for guarded dispatch and host execution

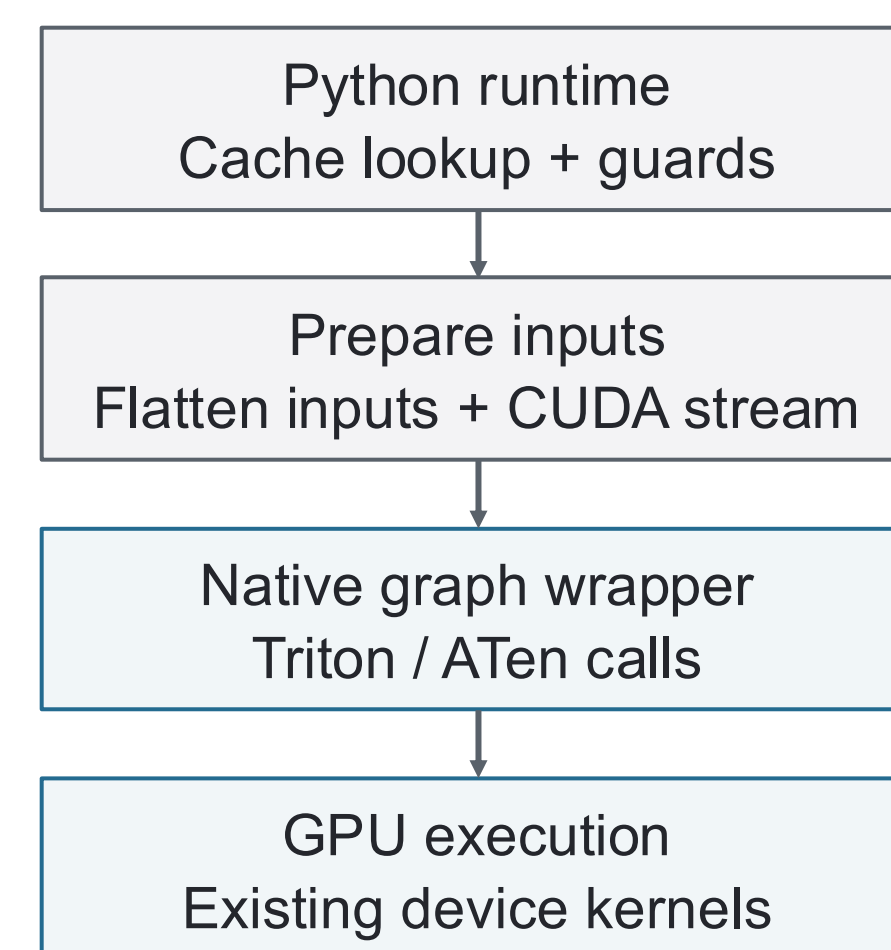
Trident compiles the recurring host path while reusing existing Triton CUBINs and ATen implementations.

URL: <https://github.com/flagos-ai/trident>

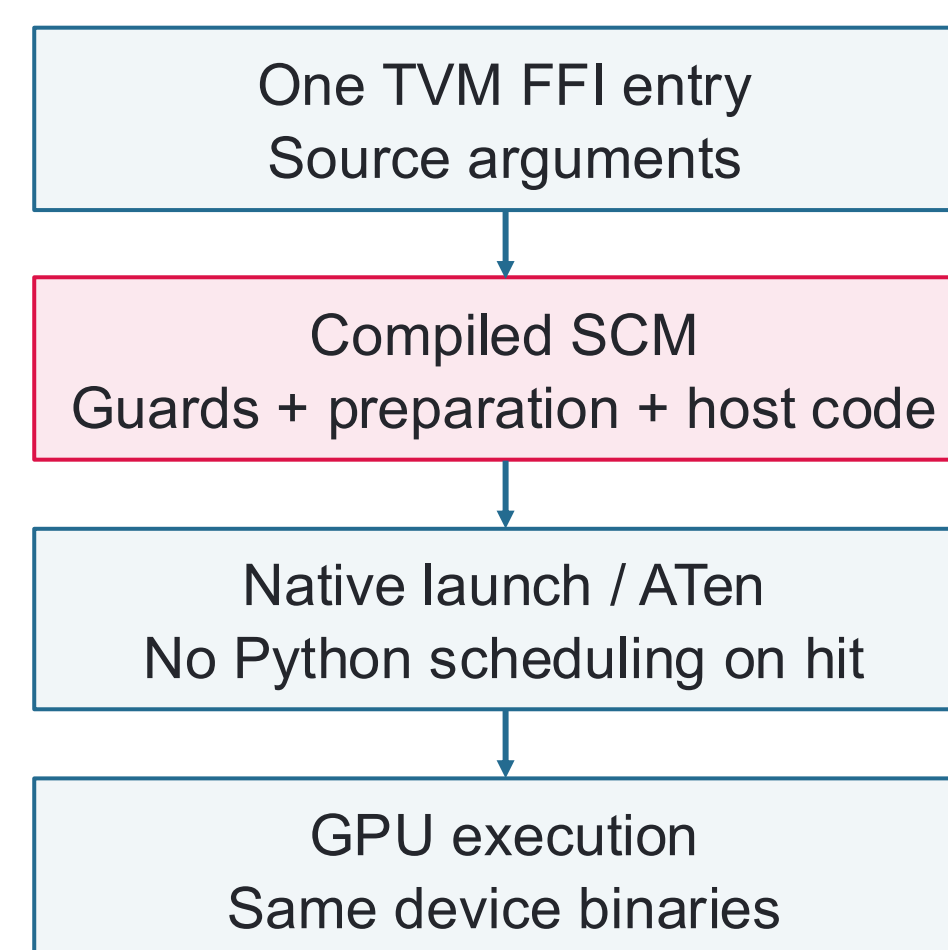
1 The recurring host path

Short GPU kernels expose the fixed cost of runtime dispatch, guards and launch preparation. A C++ graph wrapper starts after much of this work.

torch.compile



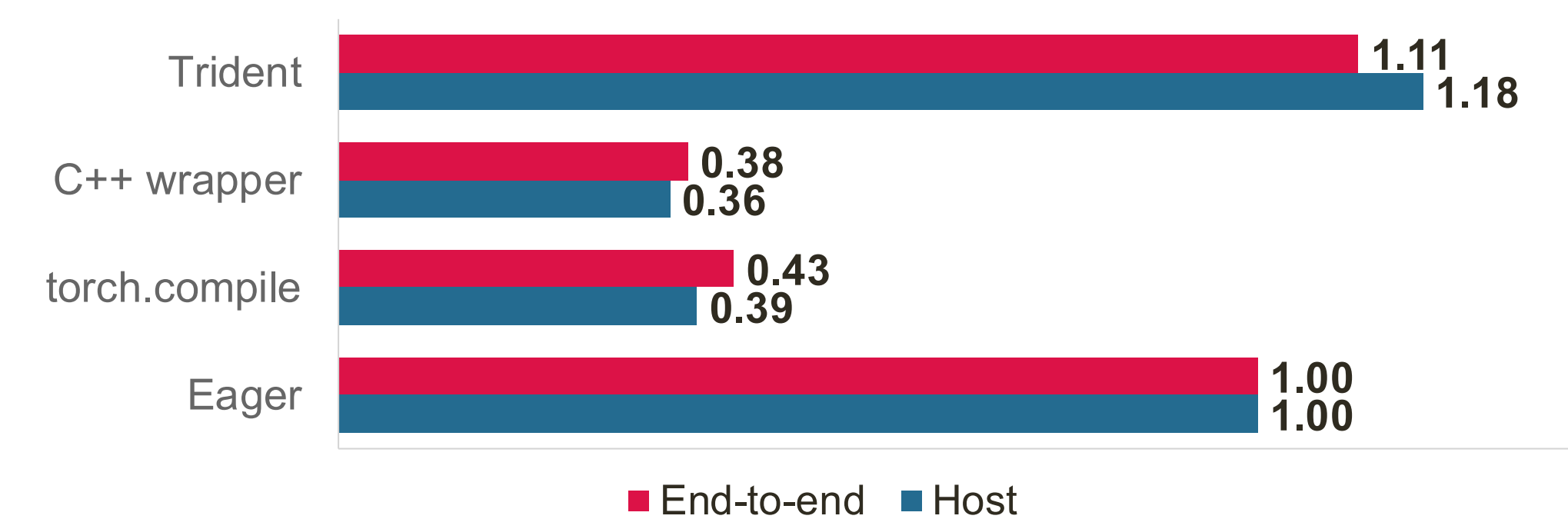
Trident warm call



Schematic runtime boundaries, not a measured timeline.
Cache hits enter native code once. Internal runtime calls remain.

rms_norm: the wrapper is not enough

Warm speedup vs eager (1.00x). Higher is faster.



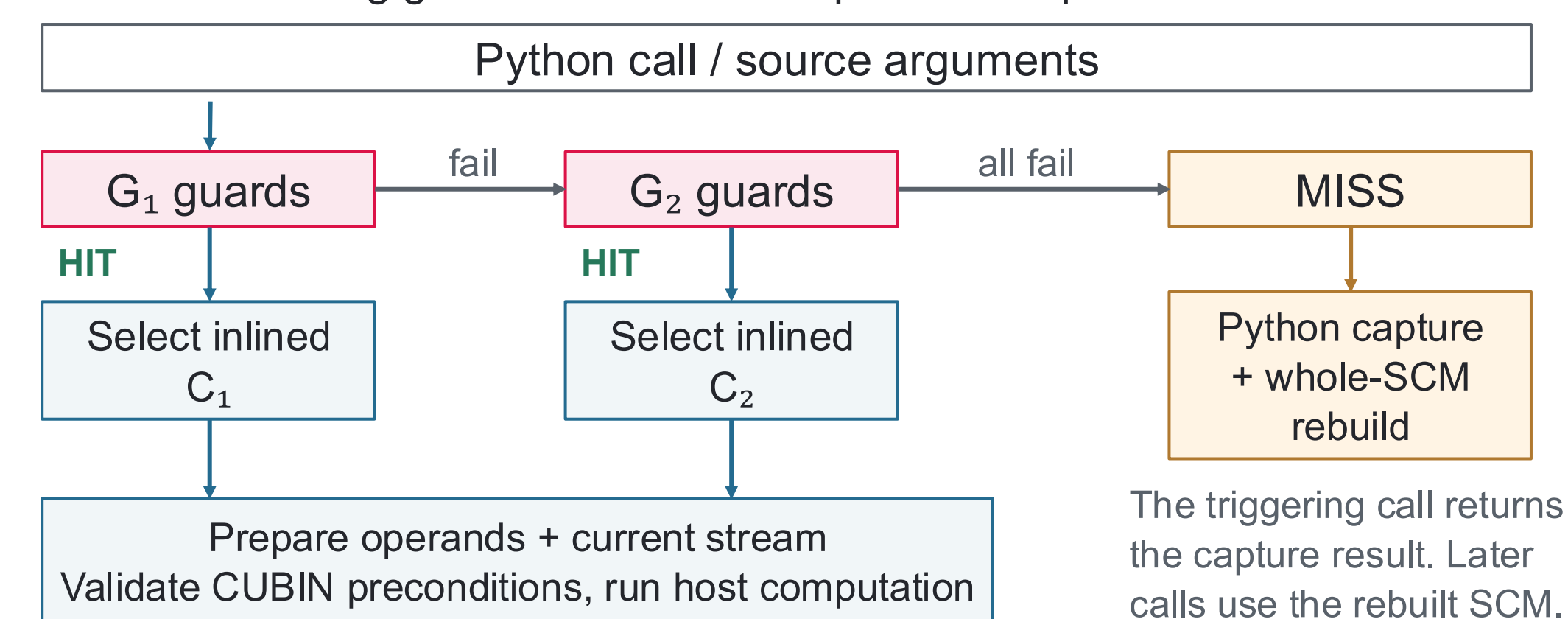
x: (1024,128), weight: (128,). 20 warmups, 10 repeats per round, 5 rounds, median. Reported gain over default torch.compile: 2.99x host and 2.60x end-to-end.

Cost of a new specialization

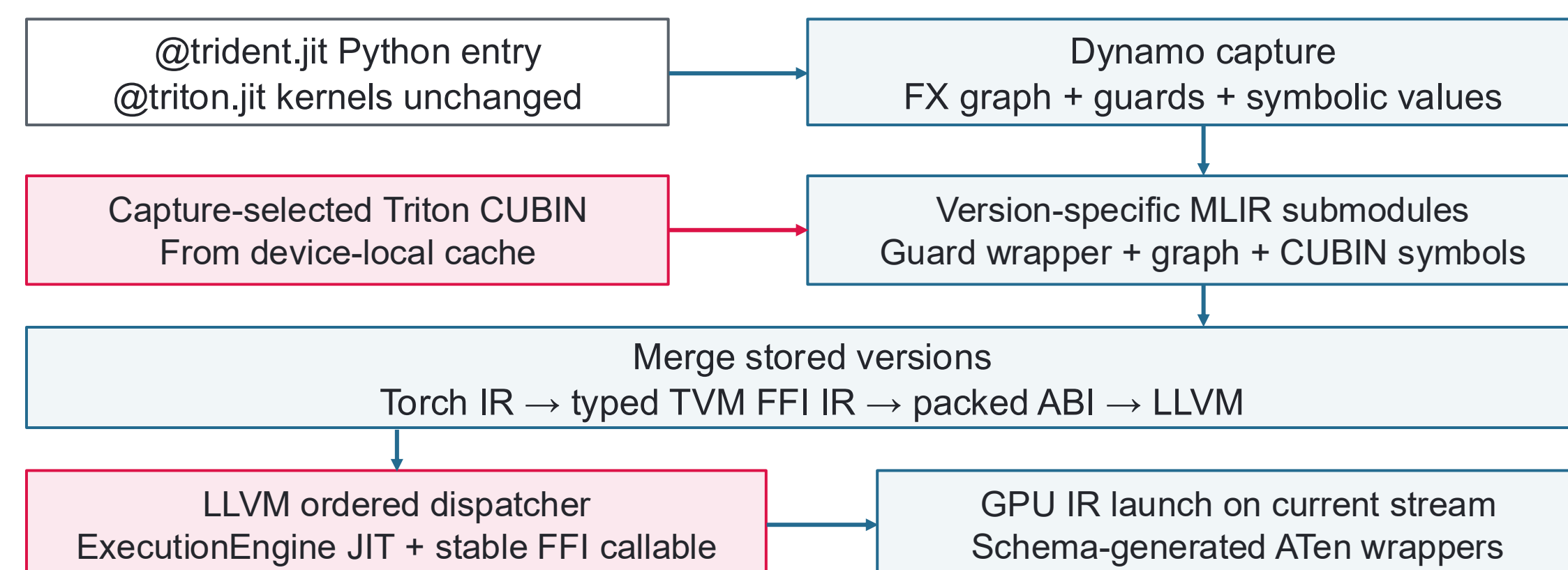
neg: 0.84 s Trident / 0.42 s torch.compile.
Cold call includes Triton compilation and SCM construction.

2 Specialization Cache Module (SCM)

Ordered versions (G_1, C_1) , (G_2, C_2) , ... share the source signature.
The first matching guard executes its captured computation.



3 Compilation and runtime reuse

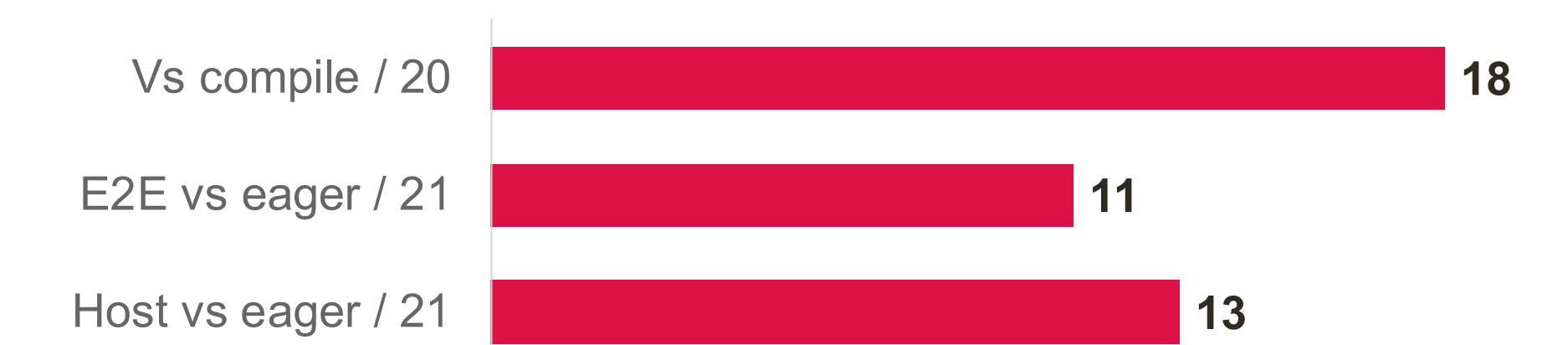


4 Experimental evidence

Single H800 80 GB, PyTorch 2.11.0
2 × Xeon Platinum 8468H (96 physical cores)

Kernel suite: broad, uneven gains

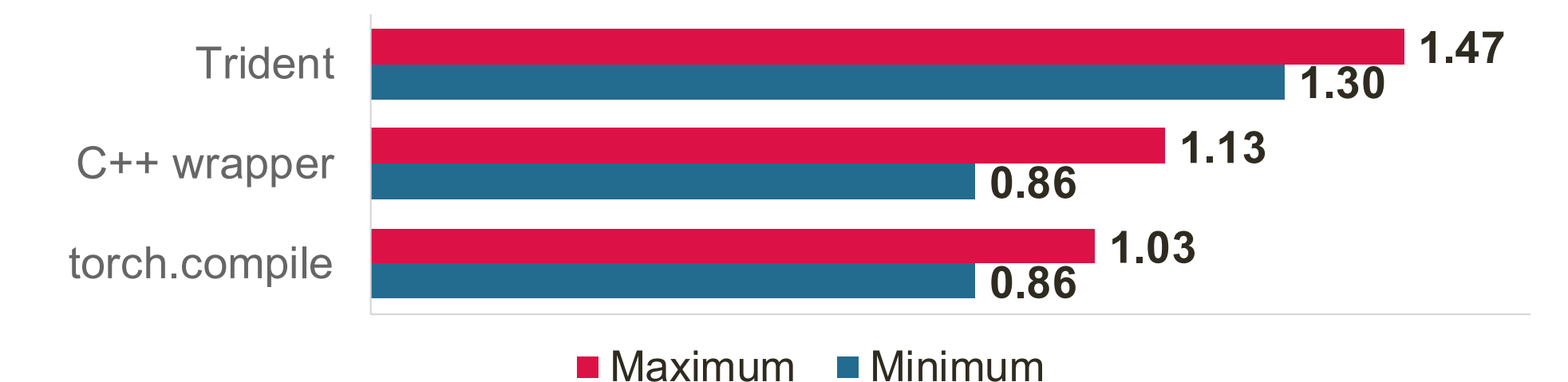
Number of operators faster than the comparison



Denominators: 21 / 21 / 20 comparable operators.
Mean vs eager: 1.10x host, 1.07x end-to-end.
Peak vs torch.compile: 3.07x host, 2.61x E2E.

Models: six model-dataset pairs

End-to-end median speedup vs eager (range)



FlagOS: open-source system software for heterogeneous AI compute. Trident is an import feature of FlagTree.

URL: flagos.io, GitHub: github.com/flagos-ai

