ISCA 2026

KernelEvolve

Scaling Agentic Kernel Coding for Heterogeneous AI Accelerators at Meta

Gaoxiang Liu, Gang Liao, and Carole-Jean Wu

On behalf of the KernelEvolve Team: Gang Liao, Hongsen Qin, Ying Wang, Alicia Golden, Michael Kuchnik Yavuz Yetim, Ruichao Xiao, Jia Jiunn Ang, Chunli Fu, Yihan He, Samuel Hsia, Zewei Jiang, Roman Levenstein, Dianshi Li, Liyuan Li, Ajit Mathews, Varna Puvvada, Feng Shi, Nathan Yan, Uladzimir Pashkevich, Matt Steiner, Carole-Jean Wu, Gaoxiang Liu

June 29, 2026
Raleigh, North Carolina USA

Meta

ISCA 2026

01 · The Problem & Current Limitations

Kernels are the translation layer at Meta scale

Trillions

ad-ranking inferences served per day

1,500+

models across a heterogeneous fleet:
NVIDIA · AMD · MTIA · CPU

The Last Mile

kernels translate model ops into chip specific instructions for each request

Kernel Efficiency

drives TCO & revenue — yet every new chip or model needs kernels rewritten and tuned

* **MTIA**: Meta Training and Inference Accelerators – https://ai.meta.com/blog/meta-mtia-scale-ai-chips-for-billions/

Meta

Explosive kernel growth

The number of kernels scales with the product in three key dimensions:

Hardware Heterogeneity

Different memory hierarchies, ISAs, and execution models, e.g., 4 MTIA generations in 2 years.

A kernel tuned for one generation underperforms on the next.

Model Architecture

DLRM → Sequence learning → Adaptive Ranking Model (LLM-scale) → Generative Recommender

New operators required per generation; single requests traverse multiple model families.

Kernel Diversity

Beyond cuBLAS/cuDNN, long tail of custom ops: hashing, bucketing, fused interactions, custom attention.

Optimal kernels vary by shape and context; no library implementation for custom ops.

Manual cost: O(operators × platforms)

2–8 weeks of expert effort per platform, invalidated every 12–18 months by new hardware.

Hand-tuning by kernel experts does not scale

Experts alone can't keep pace with the accelerating hardware and model roadmaps.

Existing approaches fall short

AI Compiler

Compilers fail behind new hardware features and new model architectures

Compilers frequently fall short on the performance

Human-in-the-loop one/few-short agent

Human become the bottleneck in finding optimal solution

Lack of systematic way to accumulate knowledge that enables cross nodes, cross session learning, and in the future post-training

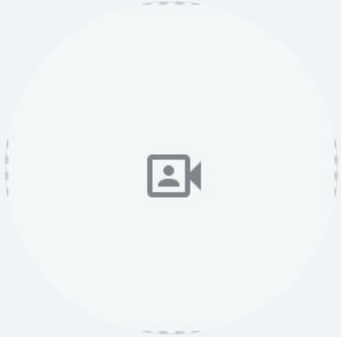
Can we develop an autonomous self evolving kernel agent that can beat SoTA kernel implementations across heterogeneous hardware

ISCA 2026

Our Proposed Solution

KernelEvolve: An Agentic Kernel Coding System

Gang Liao



Meta

ISCA 2026

02 · KernelEvolve Architecture & Execution Workflow

KernelEvolve in one picture

Our Solution

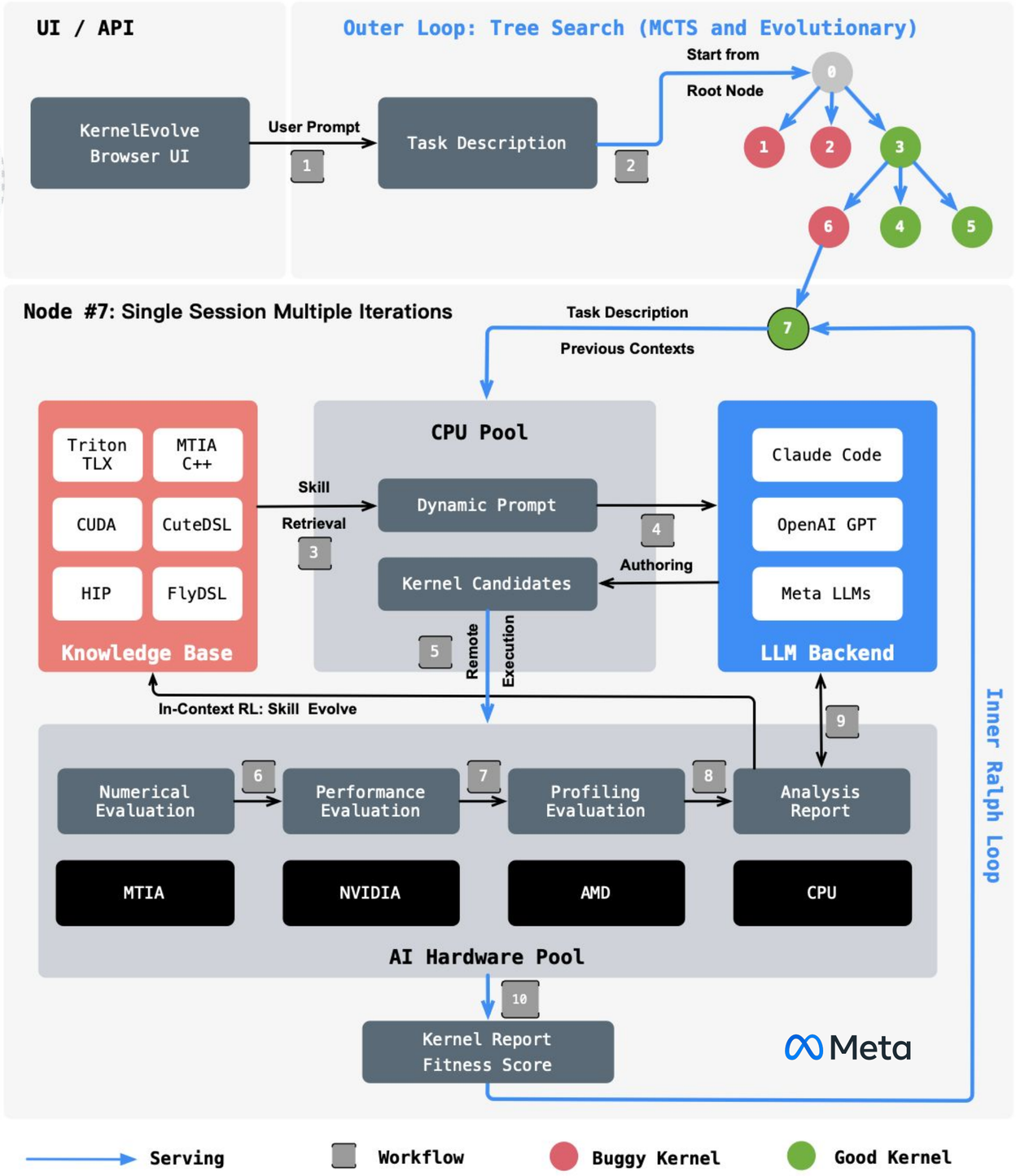
Kernel optimization reframed as **graph-based search** + **retrieval-augmented prompting** + **persistent knowledge base**.

Three Architectural Pillars:

- **Search**
- **Knowledge**
- **Production-grade Evaluation**

Feedback Loop
Diagnostics & Profiling Insights

Final Output
Kernel Artifacts



Pillar 1: Search as the Optimization Engine

01. OUTER LOOP

State machine

Decides what to explore next by selecting frontiers via the persistent exploration graph.

02. SEARCH STRATEGY

Tree Search

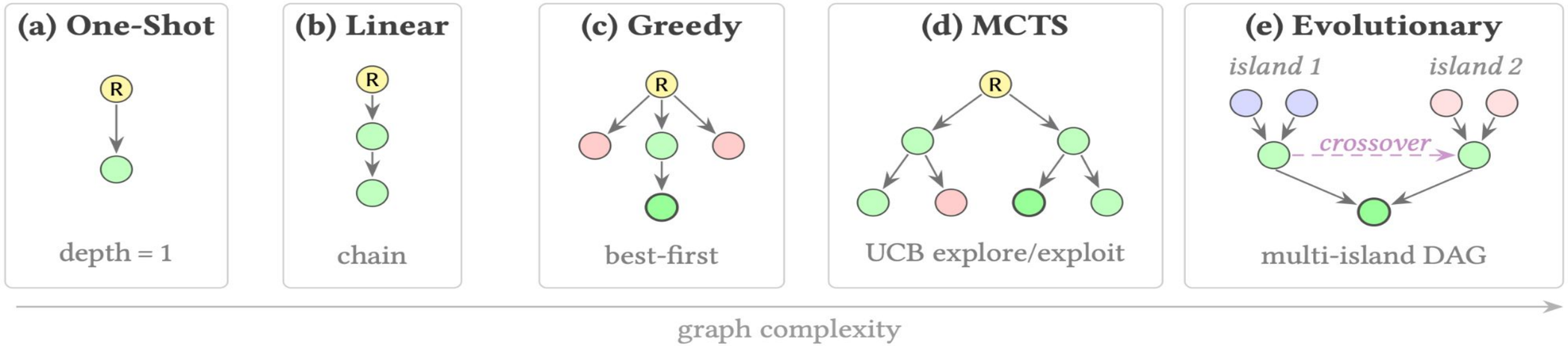
- One-Shot
- Linear
- Greedy
- MCTS
- Evolutionary Algorithms

03. FITNESS CRITERIA

Speedup vs. PyTorch

Primary metric for evaluating candidate kernel artifacts.

Hard Constraint:
Correctness failure results in a fitness of 0.



Pillar 2 — Knowledge Base for MTIA Knowledge Injection

THE CHALLENGE: PROPRIETARY SILICON GAP

MTIA is absent from LLM training corpora. A bare model emits GPU-flavored Triton that won't even compile.

HIERARCHICAL KNOWLEDGE BASE

Provides constraints and guidance for hardware targets: {NVIDIA, AMD, MTIA}.

DEEP-SEARCH by SUB-AGENTs

Retrieves profiling feedback for specialized hardware features: SFU/libdevice, cross-PE comm, dual-core, TLX warp specialization.

SCALABILITY

New hardware → inject docs, not retrain the model.

ISCA 2026

02 · KernelEvolve Architecture & Execution Workflow

Pillar 3 — Production-Grade Evaluation & Tooling

Multi-tool profiling unified

TritonBench · Torch Profiler · NCU · Proton · MTIA Insight

Triton Multi-Pass Profiler (MPP)

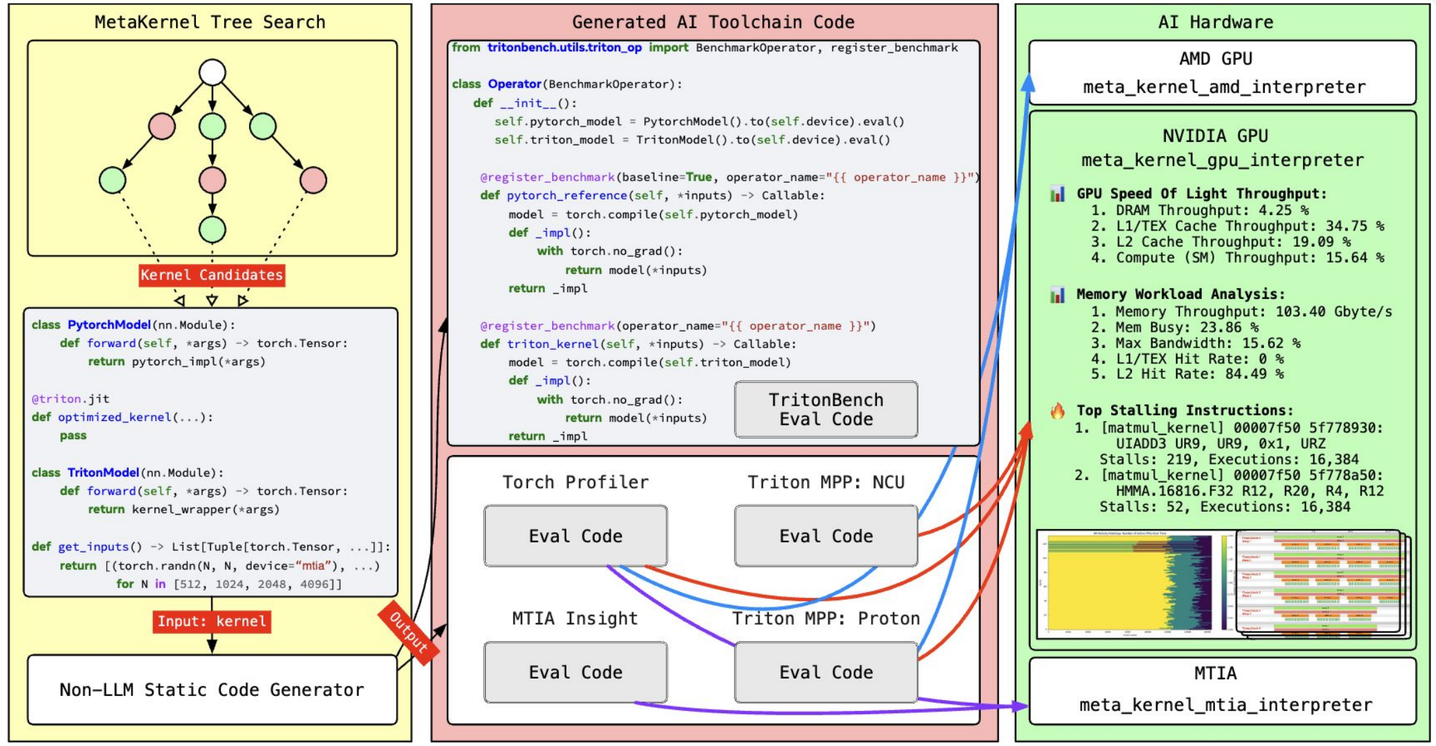
Instruction-level, minimally-invasive

FaaS evaluation

Decouples generation (CPU) from evaluation (accelerator)

JIT C++ emit + replay

Second-scale agentic debugging



This harness is what turns "an agent that writes code" into production-grade reliability (auto correctness validation + safe fallback)

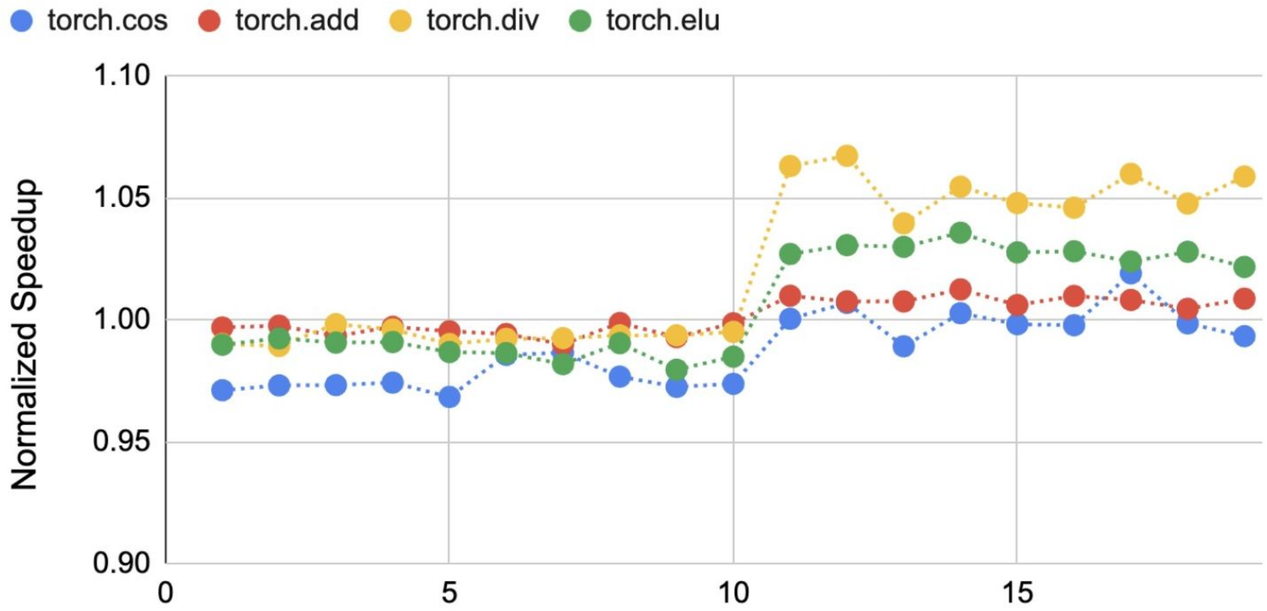
Does it work? **Correctness first**

100% Correctness

KernelBench — 100% pass rate across all OSS benchmark levels (operators, fusion, full models)

480 configs — 160 ATen operators × {H100, MI350, MTIA 400} → 100%

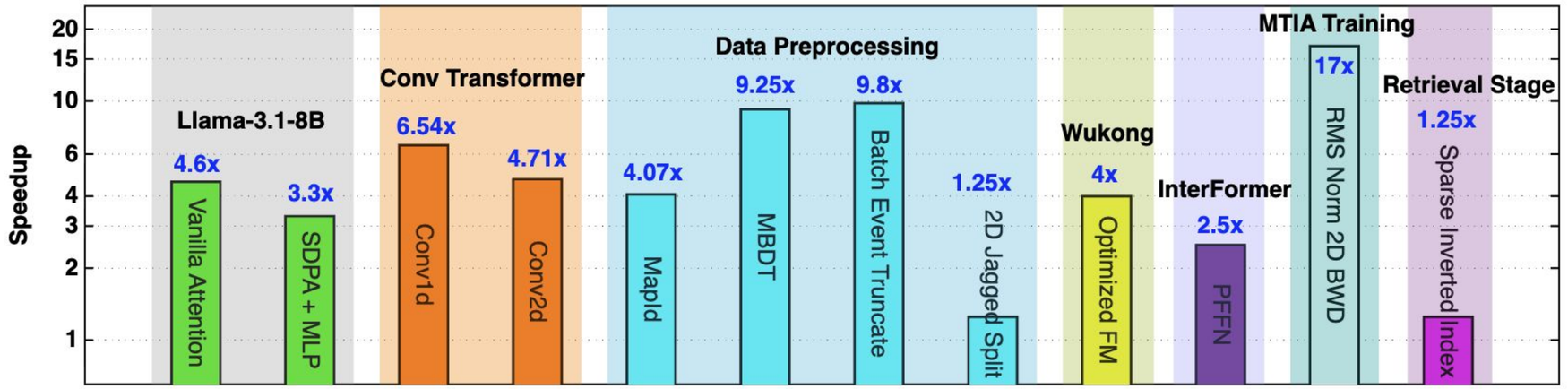
Verified via torch.allclose numerically



Correctness coverage before speedup

KernelEvolve Result Highlight

Over AI model variety

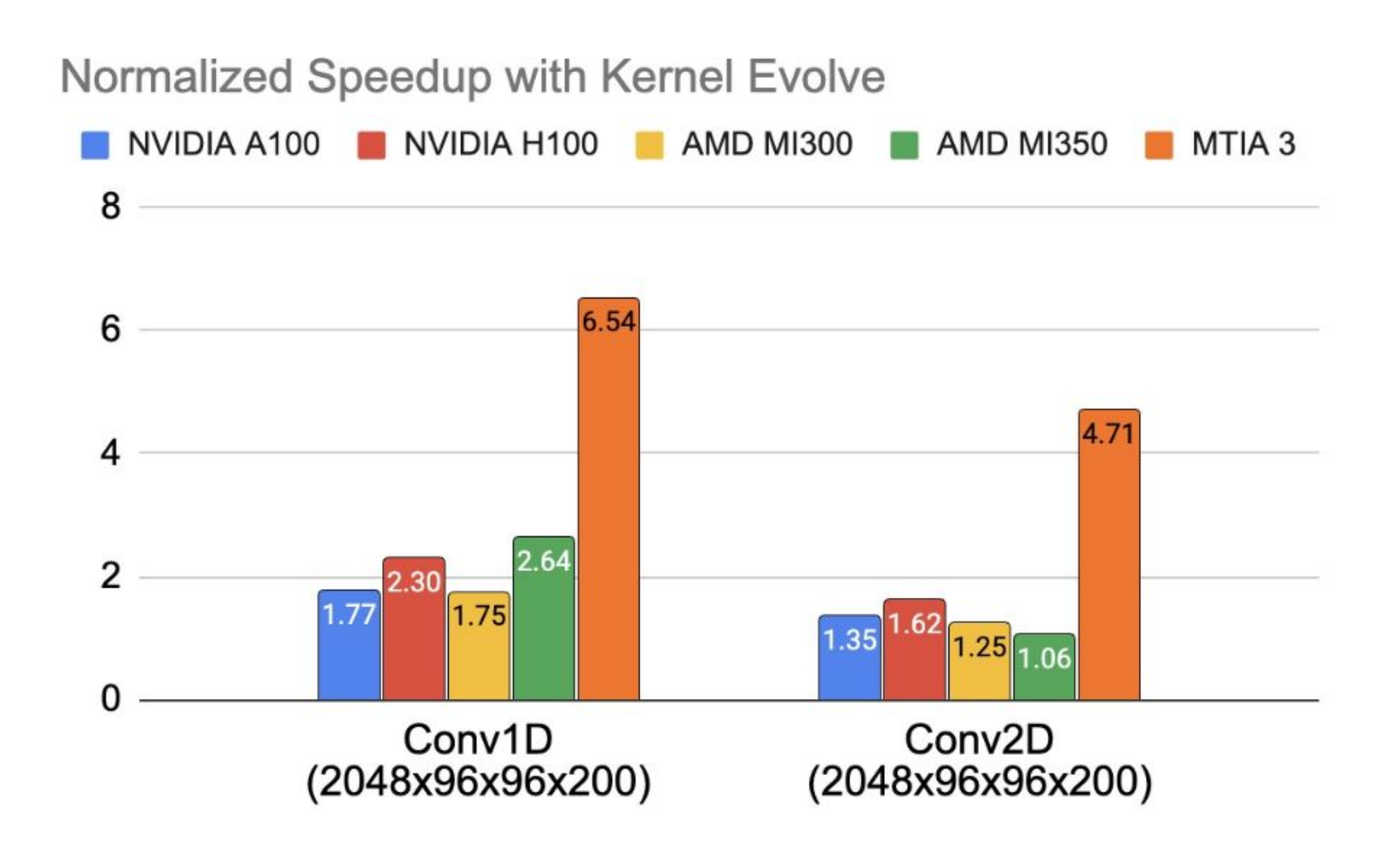


Speedup comparison across various OSS & production model architectures/workloads on MTIA Hardware

1.25–17× speedup on production workloads — kernel development drops from weeks to hours

KernelEvolve Result Highlight

Over AI hardware variety



Deep dive: where **speedup comes from**

Kernel Fusion Efficiency

conv1d: PyTorch launches 5 kernels (including layout transforms). KernelEvolve fuses these into just **2 kernels**.

Universal Hardware Support

Optimized performance across diverse architectures:

- MTIA
- NVIDIA
- AMD

Torch.Conv1d

ID	Estimated Speedup	Function Name	Demangled Na	Duration [ms] (357.41 ms)
0	17.81	nchwToNhwcKernel	..nchw2nhw...	92.64
1	526.30	nchwToNhwcKernel	..nchw2nhw...	4.70
2	45.13	sm90_xmma_fprop_implicit_gemm_bf16bf16_bf16f32_f32_nhwckrsc_...	sm90_xmm...	93.57
3	17.91	nhwcToNchwKernel	..nhwc2nch...	102.98
4	13.82	triton_poi_fused_convolution_0	triton_poi_fu...	63.52

Torch.Conv2d

ID	Estimated Speedup	Function Name	Demangled Na	Duration [ms] (224.96 ms)	Runtime Impro (84.12 ms)
0	31.69	triton_poi_fused__to_copy_unsqueeze_0	triton_poi_fu...	64.16	2
1	337.26	triton_poi_fused__to_copy_convolution_unsquee...	triton_poi_fu...	3.87	1
2	45.17	sm90_xmma_fprop_implicit_gemm_bf16bf16_b...	sm90_xmm...	93.73	4
3	13.28	triton_poi_fused__to_copy_convolution_unsquee...	triton_poi_fu...	63.20	

Triton Kernel

ID	Estimated Speedup	Function Name	Demangled Na	Duration [ms] (203.58 ms)	Runtime Impro (116.10 ms)
0	675.45	pack_conv1d_weight_kernel	pack_conv1...	6.08	4
1	37.99	conv1d_gemm_kernel	conv1d_ge...	197.50	1

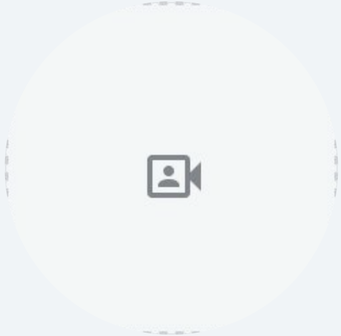
MTIA preprocessing = enablement, not just optimization. Where PyTorch falls back to CPU, KernelEvolve unblocks on-device path.

ISCA 2026

Experience & Lessons Learned

Future Directions

Carole-Jean Wu



Meta

Beyond Speedups: Agents Reveal What **Silicon Features to Build Next**

MTIA is absent from LLM training data — so the agent discovers optimizations purely through exploration, making its workarounds genuine hardware signals.

01. Local Storage Bottleneck

Search consistently maximized circular-buffer depth to cap DMA–compute overlap.

Outcome: **Next chip's Local Scratchpad increased from 512 KB to 1 MB.**

02. Inter-PE Comm Overhead

Conv1D converged on a two-phase scratch-buffer workaround due to lack of sharing primitives.

Outcome: **Hardware-native row-wise reduction introduced.**

Where the agent repeatedly works around hardware, computer architects learn what to prioritize — a scalable co-design methodology.

Our Vision

Model-Hardware Co-Degen across the Whole Stack

Cross-Stack Design

Hardware-aware codegen across the whole stack: operators → models with cross-layer fusion and end-to-end graphs.

New Abstractions

Beyond Triton to MLIR / PTX/SASS ISA code.

Identifying the right transformation layer for LLM agents.

Resource Optimization At-Scale

Optimizing across heterogeneous ISAs for fleet-wide performance and energy efficiency.

RL Execution Feedback

Adapt frontier LLMs to proprietary silicon using execution feedback without exposure.

Efficiency & Sustainability

Efficiency lowers energy; higher utilization amortizes capex and embodied carbon.

Concluding Thoughts

1. Automated Coverage

Kernel coverage is a deployment gate — and the KernelEvolve agent now provides it automatically.

2. Proprietary Optimization

Agentic search + knowledge injection → production-grade kernels for proprietary hardware like MTIA no LLM has seen.

3. Scalable Performance Wins

1.25–17× speedup with 100% correctness, serving trillions of daily inferences in production.

AI agents don't just write kernels for our hardware — they inspire us with what hardware to build next.

More details:
https://engineering.fb.com/2026/04/02/developer-tools/kernelevolve-how-metas-ranking-engineer-agent-optimizes-ai-infrastructure/

Meta


Meta