MICRO 2026
59th IEEE/ACM International Symposium on Microarchitecture



FlashGPU-sim: Enabling GPU Modeling for Modern Architectures and AI Workloads

Siying Yu[1], Yixun Hong[1,3], Guozhi Qiu[1,3], Jingci Liu[1], Feng Gu[1], Chenbo Geng[2], Zhengrong Wang[1], Chen Zhang[2], Bei Yu[1]

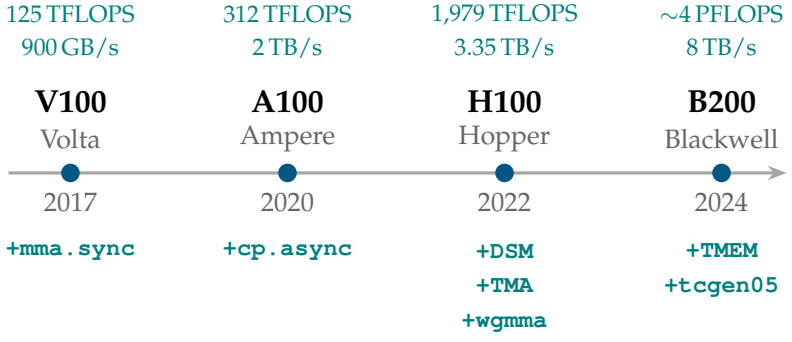
[1] Chinese University of Hong Kong
[2] Shanghai Jiao Tong University
[3] Zhejiang University

Nov. 2, 2026



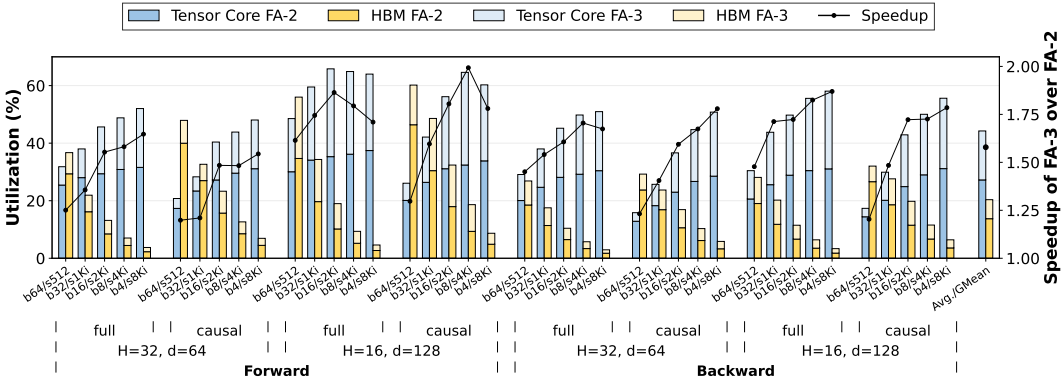
The AI–Hardware Specialization Flywheel

AI's ever-increasing demand for compute drives hardware toward greater specialization:



The Accelerating Flywheel: Specialized hardware $\longleftrightarrow$ Tailored AI kernels

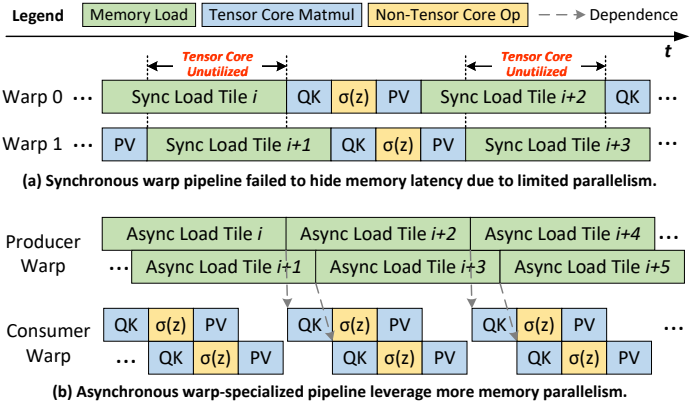
The Flywheel in Action: FlashAttention on H100



FA-3 vs. FA-2: 1.58× geometric-mean speedup, up to 1.99×

FA-3 specializes its execution for Hopper's asynchronous hardware.

How to Achieve This: Async Execution Reorganization



Each row represents one warp; time flows from left to right.

FA-3 builds a specialized async pipeline:

- **TMA**: async data movement
- **WGMMA**: async tensor compute
- **Warp specialization**: producers move data; consumers compute

Reducing coupling between resources enables sustained higher throughput.

The Architect's Question

How can we study future GPU designs for evolving AI kernels?

- Use Old Simulators?
- Profiling on Existing Hardware?

Simulator	Year	Target Platform	Latest Arch	Arch Release	Method	Modern AI Workloads	Multi-thread Acceleration
GPGPU-Sim	2009	NVIDIA	Volta	2017	Execution	✘	✘
Multi2Sim	2012	Both	Kepler	2012	Execution	✘	✘
gem5-gpu	2015	NVIDIA	Fermi	2010	Execution	✘	✘
MGPUSim	2019	AMD	GCN3	2016	Execution	✘	✓
Accel-Sim	2020	NVIDIA	Ampere	2020	Trace & Execution	✘	✘
Huerta et al.	2025	NVIDIA	Ampere	2020	Trace & Execution	✘	✓
FlashGPU-sim	**2026**	**NVIDIA**	**Blackwell**	**2024**	**Execution**	✓	✓

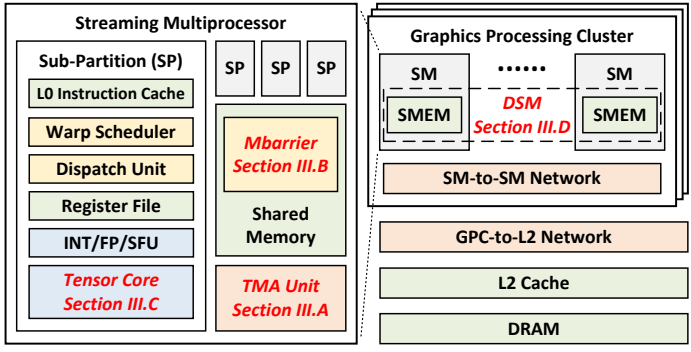
FlashGPU-sim fills this gap:

A credible platform to study the flywheel between AI workloads and hardware.

FlashGPU-sim

System Overview

Execution-driven, cycle-accurate modeling, extended from GPGPU-Sim.



- **TMA**: async data movement
- **mbarrier**: fine-grained synchronization
- **Tensor Cores**: mma, wgmma, tcgen05/TMEM
- **DSM**: cross-SM communication

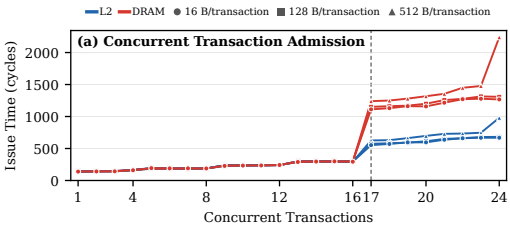
- **Triton frontend**: Extract and replay production AI kernels without manual porting.
- **Multi-threaded acceleration**: practical simulation at AI-kernel scale.

Calibration via Reverse Engineering

NVIDIA documents ISA interfaces – **not microarchitectural timing**.

Execution Primitives

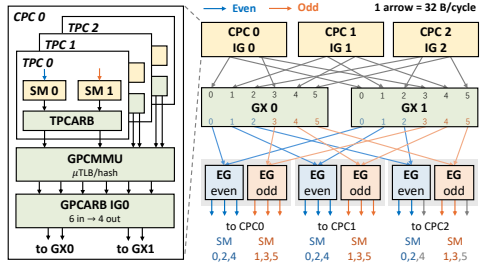
TMA · MMA · mbarrier



Latency · Throughput · Resource Constraints

Inter-SM Communication

Distributed Shared Memory

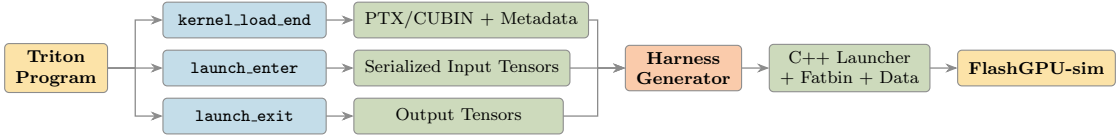


Topology · Routing · Contention

Hardware-informed modeling and calibration of key GPU mechanisms.

Triton Frontend

Workflow: Autotune · Capture · Replay



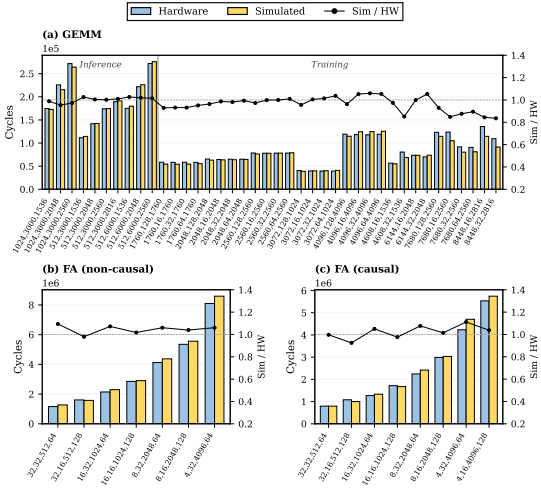
Example

```
import TritonTrace
tracker = TritonTrace.Tracker(
    output_dir, mode="online", enabled=False)
my_kernel[grid](A, B, C, M, N, K) # autotune
tracker.enable()
my_kernel[grid](A, B, C, M, N, K) # capture
tracker.save_summary()
```

Key Features

- **Only 3 lines** of Python to instrument any Triton kernel.
- Supports **production Triton kernels** e.g. FlashAttention, fused MoE, ...

Validation with AI Workloads

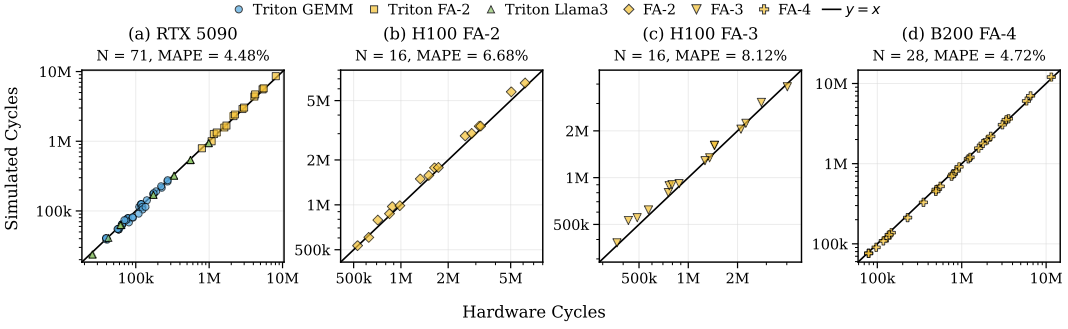


GEMM and FlashAttention kernel validation

Llama3-8B layer-level inference validation

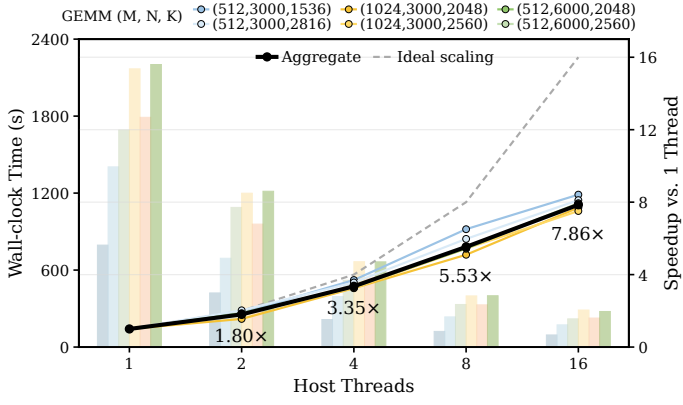
#	Kernel	Sim	NCU	Diff	Error
Prefill: batch 2, sequence length 128					
1	RMSNorm	62,794	62,044.7	+749	+1.2%
2	Linear	319,315	332,203.6	−12,889	−3.9%
3	GQA	41,090	42,427.2	−1,337	−3.2%
4	Linear	170,275	173,518.3	−3,243	−1.9%
5	RMSNorm	62,794	63,653.1	−859	−1.3%
6	Linear	940,773	988,824.6	−48,052	−4.9%
7	SwiGLU	23,672	25,995.3	−2,323	−8.9%
8	Linear	537,878	557,109.7	−19,232	−3.5%
	Total	*2,158,591*	*2,245,776.3*	*−87,185*	*−3.9%*
Decode: batch 256, query length 1, KV length 128					
1	RMSNorm	62,794	63,148.6	−355	−0.6%
2	Linear	320,897	333,182.4	−12,285	−3.7%
3	GQA	537,986	557,999.7	−20,014	−3.6%
4	Linear	170,275	174,426.6	−4,152	−2.4%
5	RMSNorm	62,794	61,592.0	+1,202	+2.0%
6	Linear	940,773	987,916.0	−47,143	−4.8%
7	SwiGLU	23,672	25,512.5	−1,840	−7.2%
8	Linear	537,878	557,018.0	−19,140	−3.4%
	Total	*2,657,069*	*2,760,795.7*	*−103,727*	*−3.8%*

Cross-Architecture Cycle Correlation



- 131 configurations on RTX 5090, H100, and B200
- 5.24% overall MAPE.

Multi-Threaded Acceleration

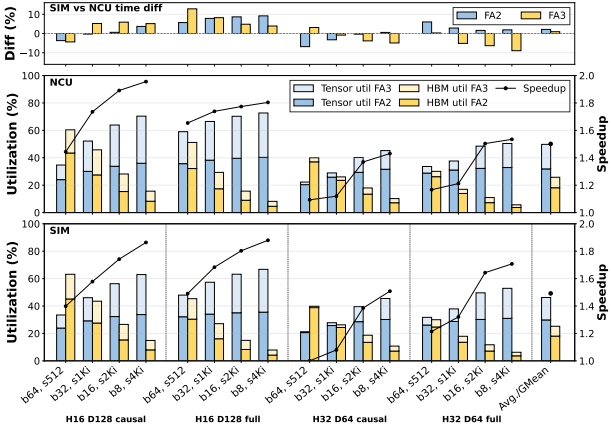


- **Test setting**: AMD EPYC 9115; host threads pinned to distinct physical cores.
- **16 host threads:** 7.86× speedup, turning a day-long run into just 3 hours.

Architecture Exploration

Case Study: FlashAttention on H100

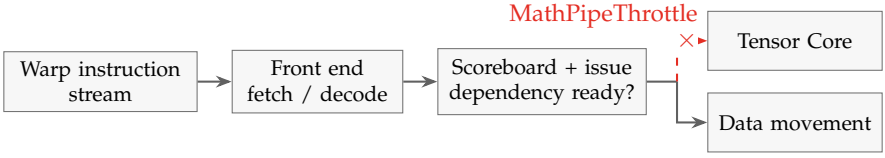
What limits FA-2 on H100, and **how can we improve it** architecturally?



Reproduce → Diagnose → Explore

- 16 complete FA-2/FA-3 pairs across all shapes.
- Average timing error $< 3\%$; maximum $\approx 13\%$.
- Simulator preserves the performance trends.

Bottleneck: Math Issue Pressure

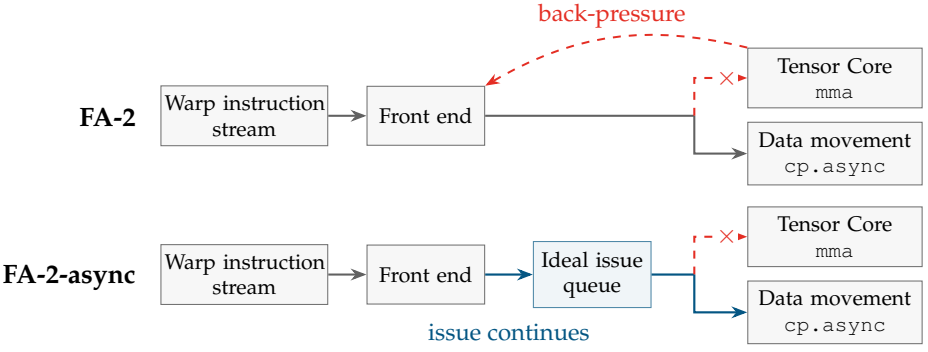


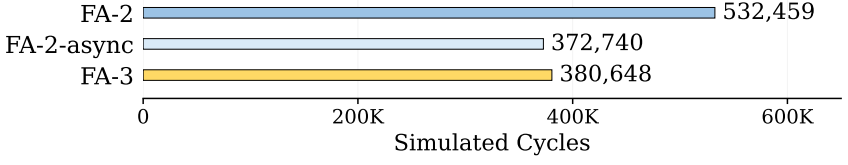
Dynamic Instruction Counts

	FA-2	FA-3
Tensor	20.97M `mma.sync`	0.573M `wgmma`
Data movement	1.57M `cp.async`	2.3K TMA
Shared matrix	6.55M `ldmatrix`	0.262M `stmatrix`
Total warp inst.	93.23M	69.76M

- FA-2's fine-grained instructions create heavy front-end pressure.
- `MathPipeThrottle` accounts for 41.1% of stalls.

What-if: Issue-Execution Decoupling





Architectural Insights



Our Finding	Need for decoupled asynchronous execution 1.43× speedup with ideal issue queue
GPU Evolution	Increasingly coarse-grained asynchronous primitives Hopper (`wgmma` / TMA) → Blackwell (`tcgen05` / TMEM)

FlashGPU-sim makes this analysis systematic and repeatable

- Sub-kernel stall breakdown for any kernel configuration
- What-if on mechanisms that don't exist in silicon yet
- Enables architects to validate design intuitions before tapeout

Conclusion



FlashGPU-sim[1]: An open-source platform:

- ✓ Cycle-accurate modeling: TMA, Tensor Core, DSM, ...
- ✓ Triton Frontend
- ✓ 5.24% MAPE across RTX 5090, H100, and B200
- ✓ 7.86× multi-threaded speedup

Development Roadmap

- Multi-GPU + NVLink modeling
- gem5 integration

Building on FlashGPU-sim

- Architecture exploration
- Faster and more scalable simulation
- Agentic workflow integration
- ...

[1] **https://github.com/FlashGPU-Sim/FlashGPU-Sim**

THANK YOU!