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

3B.1

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

Introduction

AI and GPU hardware evolve in a reinforcing flywheel: growing AI demands drive hardware specialization, while new GPU capabilities enable tailored kernels that advance AI performance.

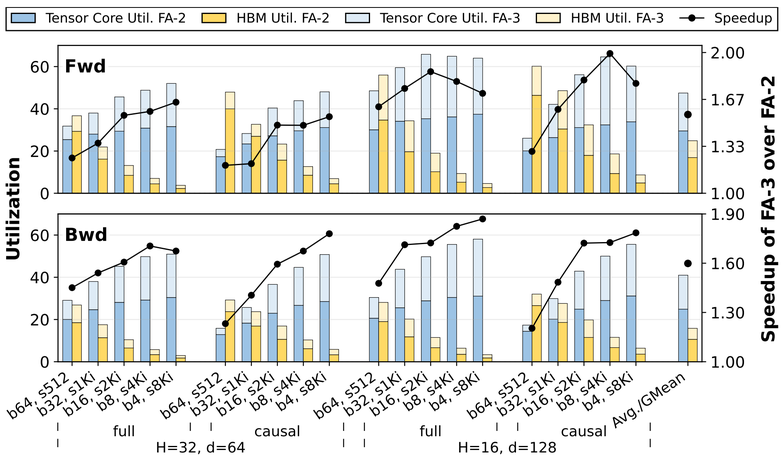


Figure 1. FA-3/FA-2: 1.58× geomean, 1.99× peak speedup.

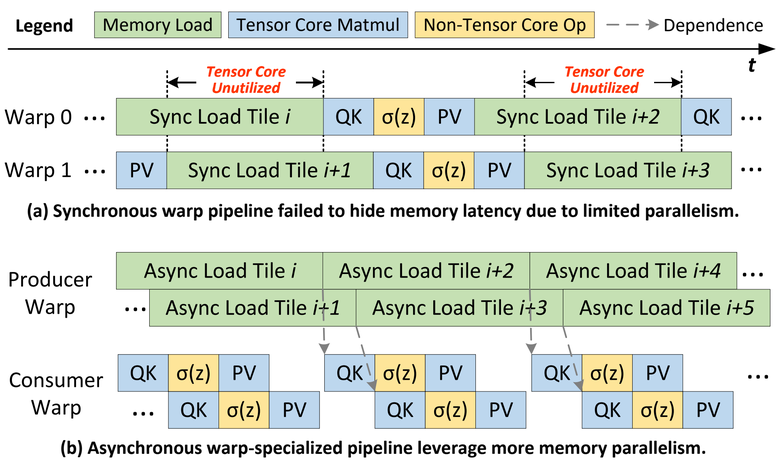


Figure 2. FA-3 overlaps data movement and tensor computation.

Research gap. Existing simulators largely target pre-Hopper architectures; hardware profiling cannot evaluate future designs.

FlashGPU-sim. An open-source, execution-driven, cycle-accurate simulator supporting Hopper and Blackwell features. Code available at `https://github.com/FlashGPU-Sim/FlashGPU-Sim`.

5.24%	7.86×	131
cycle MAPE (vs. NCU)	simulation speedup (16 threads)	validated configs (3 GPU platforms)

Modern GPU Mechanisms

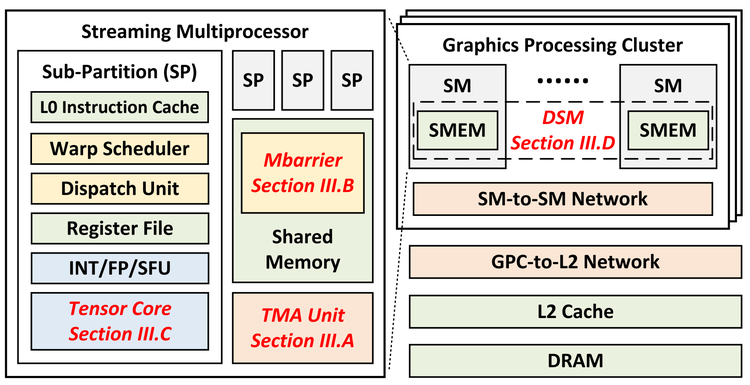


Figure 3. FlashGPU-sim architecture.

We characterize TMA, mbarrier, and warp-level MMA on RTX 5090, asynchronous Tensor Cores on H100/B200, and DSM on H200.

TMA offloads bulk data transfers to shared memory.

- **Issue gap:** 44 cycles within one warp.
- **Admission:** up to 16 active transactions.
- **Read rate:** 32 B/SM/cycle; B200: 128 B/SM/cycle.
- **Tracking:** ≥814 requests/SM (32 B each).

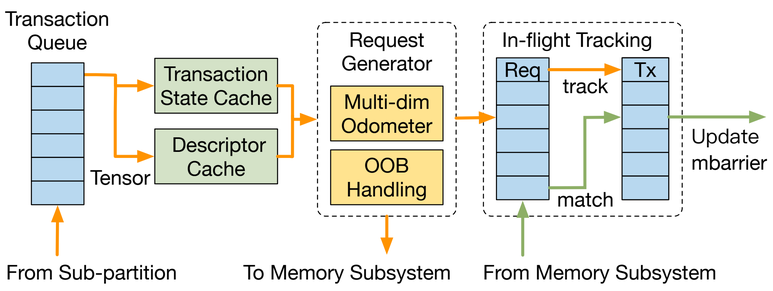


Figure 4. TMA request generation, tracking, and completion.

mbarrier synchronizes asynchronous pipeline stages.

- **Polling latency:** 42 cycles without a suspend hint.
- **Hinted wait:** 150 cycles even when already complete.

Tensor Cores accelerate matrix multiply-accumulate operations.

- **FP16/FP32 `mma.sync`:** H100/B200: ∼66%/∼25% of peak.
- **Interference:** WGMMA slows concurrent softmax on H100.

DSM enables shared-memory access across SMs.

- **Per-SM rate:** 20–21 B/cycle on H200.
- **Bandwidth allocation:** fixed per-SM quotas; no redistribution.
- **Bidirectional reads:** per-direction throughput falls 23%.

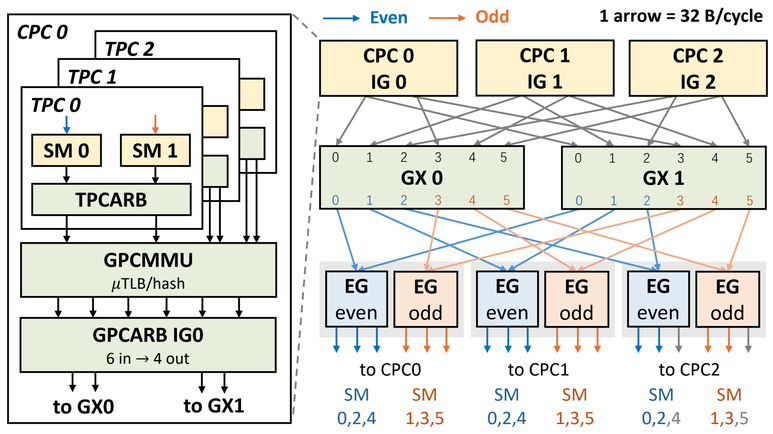


Figure 5. Intra-GPC DSM fabric.

Triton Frontend

Simulate optimized Triton kernels without manual porting.

- **Autotune:** select the best kernel configuration.
- **Capture:** save code, tensors, and launch metadata.
- **Replay:** generate a standalone simulation harness.

```python
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()
```

Figure 6. **Only 3 lines** of Python to instrument any Triton kernel.

Validation

Setup. RTX 5090; hardware cycles from `ncu`. SM clock: 2.58 GHz (Llama3: 1.80 GHz).

AI kernels: 4.8% MAPE across 40 GEMMs + 15 FA configs.

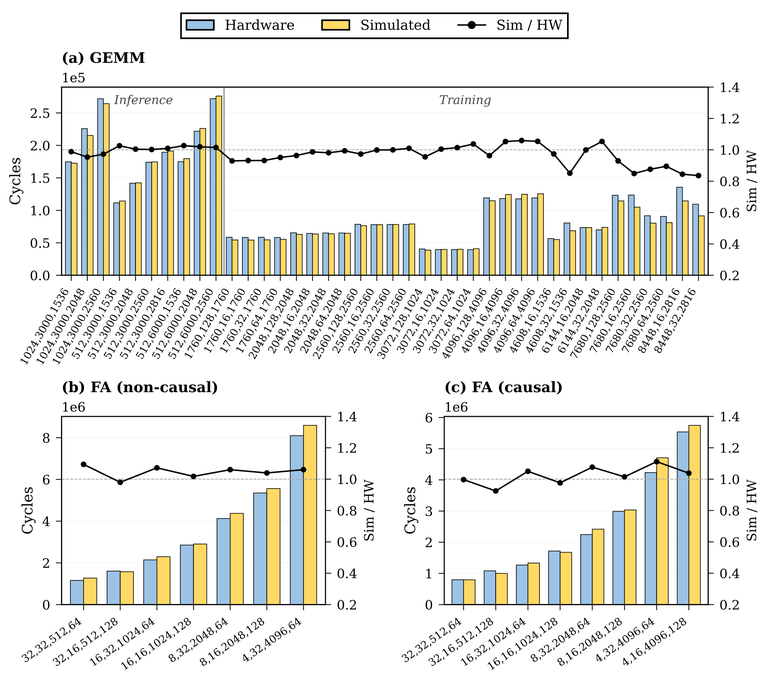


Figure 7. GEMM and FlashAttention cycle validation.

Llama3-8B: 3.5% MAPE for layer-level prefill/decode.

#	Kernel	Function	Sim	NCU	Diff (cy)	Diff (%)
Prefill: batch 2, sequence length 128						
1	RMSNorm	Input normalization	62,794	62,044.7	+749	+1.2%
2	Linear	QKV projection	319,315	332,203.6	−12,889	−3.9%
3	GQA	Causal prefill attention	41,090	42,427.2	−1,337	−3.2%
4	Linear	Output projection + residual	170,275	173,518.3	−3,243	−1.9%
5	RMSNorm	FFN normalization	62,794	63,653.1	−859	−1.3%
6	Linear	FFN up/gate projection	940,773	988,824.6	−48,052	−4.9%
7	SwiGLU	SiLU + gate multiply	23,672	25,995.3	−2,323	−8.9%
8	Linear	FFN down projection + residual	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	Input normalization	62,794	63,148.6	−355	−0.6%
2	Linear	QKV projection + KV-cache write	320,897	333,182.4	−12,285	−3.7%
3	GQA	Single-token decode attention	537,986	557,999.7	−20,014	−3.6%
4	Linear	Output projection + residual	170,275	174,426.6	−4,152	−2.4%
5	RMSNorm	FFN normalization	62,794	61,592.0	+1,202	+2.0%
6	Linear	FFN up/gate projection	940,773	987,916.0	−47,143	−4.8%
7	SwiGLU	SiLU + gate multiply	23,672	25,512.5	−1,840	−7.2%
8	Linear	FFN down projection + residual	537,878	557,018.0	−19,140	−3.4%
	Total		*2,657,069*	*2,760,795.7*	*−103,727*	*−3.8%*

Cross-architecture validation.

- **H100:** FA-2/FA-3 MAPE 6.68%/8.12% (16 configs each).
- **B200:** FA-4 MAPE 4.72% (28 configs).
- **Overall:** 5.24% MAPE across 131 configs on 3 GPUs.

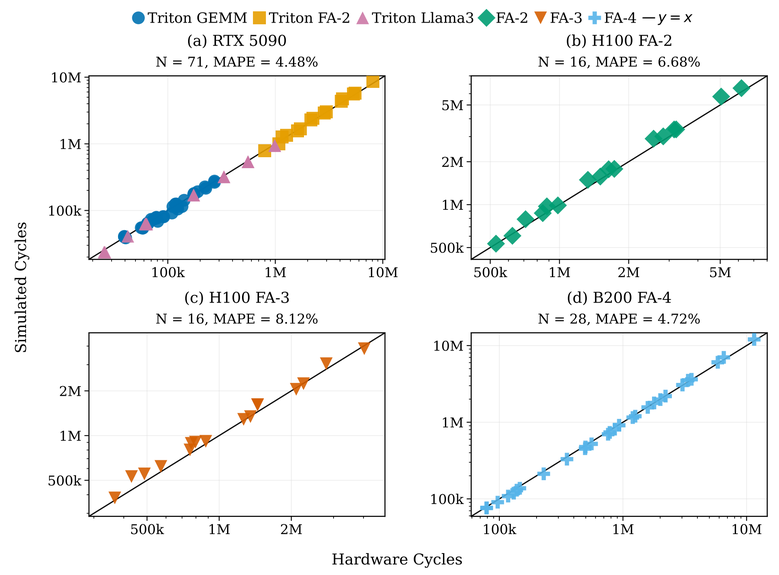


Figure 8. Cross-architecture cycle correlation.

Simulation speed: 7.86× with 16 threads (EPYC 9115).

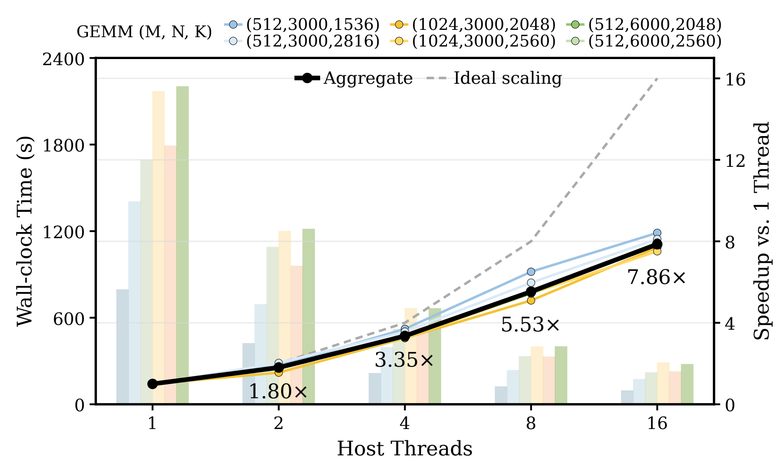


Figure 9. Multi-threaded simulation speedup.

Case Study

What limits FA-2 on H100? Reproduces FA-2/FA-3 performance trends with <5% mean timing error.

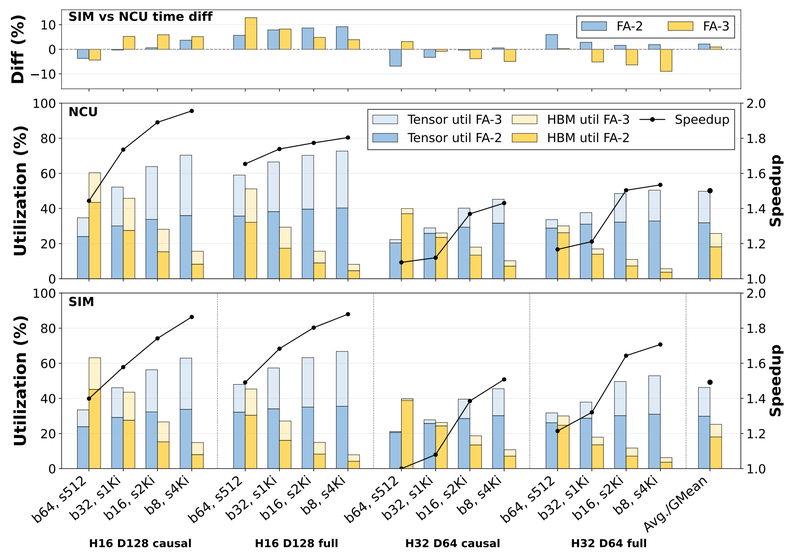


Figure 10. FA-2/FA-3 on H100: simulation vs. hardware.

Representative case: B64, H16, S512, D128, causal.

Dynamic instructions	FA-2	FA-3
Tensor compute	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 instructions	93.23M	69.76M

- **Bottleneck:** MMA issue throttling causes 41.1% of FA-2 stalls.
- **Occupancy:** 2 → 3 CTAs/SM yields no speedup.
- **What-if:** ideal queue between issue and MMA/data execution.

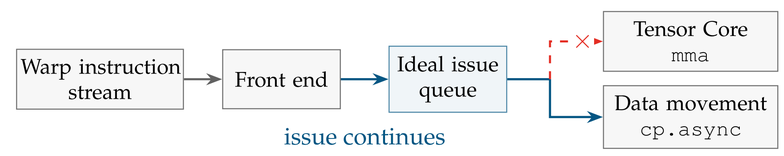


Figure 11. Issue–execution decoupling.

Result: 30% fewer cycles for FA-2, only 2.1% below FA-3.

Conclusion. Fine-grained instructions create front-end issue pressure; decoupling issue from execution improves overlap.

FlashGPU-sim makes the bottleneck analysis and architectural exploration systematic and reproducible.

59th IEEE/ACM International Symposium on Microarchitecture, Athens, Greece