A Survey of High-Level Modeling and Simulation Methods for Modern Machine Learning Workloads

MICRO 2026 Submission – Confidential Draft – Do NOT Distribute!!

Anonymous Author(s)
Under Review
Anonymous

Abstract

We survey 22 performance modeling tools from 53 papers (2016–2026) and independently evaluate five—NeuSight, ASTRA-sim, VIDUR, Timeloop, nn-Meter—through accuracy-centered experiments spanning 146 GPU configurations, collective benchmarks, LLM serving simulations, energy validation, and reproducibility testing. Three findings emerge. First, self-reported accuracy is unreliable: NeuSight claims 2.3% MAPE but we measure 5.87–27.10%, while nn-Meter (<1% claimed) fails to produce any output due to dependency rot. Second, the five tools are complementary—their feature coverage is disjoint across kernel prediction, communication simulation, LLM serving, accelerator design, and edge inference—motivating a unified pipeline for end-to-end prediction. Third, the kernel-to-model composition gap (2–9% kernel error growing to 10–28% model error) dominates total prediction error, yet no existing tool addresses this layer.

Keywords

ML workload performance prediction, DNN accelerator modeling, GPU simulation, distributed training simulation, LLM inference serving, design space exploration, survey

1 Introduction

Machine learning workloads have become the dominant consumers of compute across datacenters and edge devices. Training and inference for CNNs, transformers, mixture-of-experts models, and LLMs demand hardware ranging from Google's TPU [34, 35] to custom accelerators, creating a heterogeneous landscape where architects must predict performance before committing to costly hardware decisions.

The shift toward domain-specific architectures [25] makes performance prediction both more important and more difficult. Design space exploration, parallelization selection, and hardware-software co-design all require fast, accurate performance models—yet ML workloads pose unique challenges: diverse computational patterns (dense matrix operations, sparse accesses, communication-bound collectives) across GPUs, TPUs, custom accelerators, and multi-device clusters.

A rich ecosystem of modeling tools has emerged. Analytical models (Timeloop [56], MAESTRO [42]) evaluate in microseconds with 5–15% error. Trace-driven simulators (ASTRA-sim [82], VIDUR [3]) replay execution traces for system-level modeling. Hybrid approaches (NeuSight [47]) combine analytical structure with learned components. Yet no prior work examines *why* certain modeling approaches

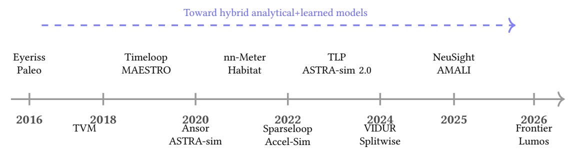

Figure 1: Evolution of performance modeling tools (2016–2026). Early analytical frameworks gave way to systematic accelerator modeling and distributed training simulation. Recent work targets LLM-specific and hybrid approaches.

succeed on certain platforms, or how prediction errors propagate across the abstraction stack. Existing surveys focus on ML *techniques* for modeling [74] or specific hardware [56]; this survey goes beyond cataloging tools to identify cross-cutting architectural principles that explain when and why different approaches work.

We make four contributions:

- **Accuracy-centered independent evaluation** of five tools using our own experiments (146 GPU configurations, collective benchmarks, LLM serving simulations, energy validation), revealing that self-reported accuracy claims are overstated by 2–4× and entirely unverifiable for the tool claiming the lowest error (Section 7).
- A **feature availability matrix** showing that the five tools cover fundamentally disjoint slices of the ML performance stack—no two tools compete on the same prediction task (Section 7).
- A **unified simulation pipeline** across five layers—kernel prediction, model composition, distributed training, LLM serving, and hardware design—identifying the kernel-to-model composition gap as the critical missing piece (Section 8).
- A **coverage matrix** exposing structural research gaps, with a **research agenda** centered on composition modeling, unified input formats, cross-hardware transfer, and continuous validation (Sections 4, 9).

Figure 1 illustrates the evolution of performance modeling tools from early analytical frameworks to modern hybrid approaches.

2 Survey Methodology

We searched ACM Digital Library, IEEE Xplore, Semantic Scholar, and arXiv using terms related to ML performance modeling, with backward/forward citation tracking from seminal works. Target venues include architecture (MICRO, ISCA, HPCA, ASPLOS), systems (MLSys, OSDI, SOSP, NSDI), and related (NeurIPS, MobiSys,

MICRO 2026, Austin, TX, USA
2026.

DAC, ISPASS). Papers must propose or evaluate a tool for predicting ML workload performance with quantitative evaluation; we exclude non-performance tasks and general-purpose workloads. From 287 initial candidates, title/abstract screening yielded 118 papers; full-text review reduced the set to 53 that met all criteria, supplemented by 12 foundational works for context. We cover 2016–2026 and classify each paper by *methodology type* (analytical, simulation, trace-driven, ML-augmented, hybrid), *target platform*, and *abstraction level* (kernel, model, system).

Related surveys and scope boundaries. Prior surveys address adjacent topics: Rakhshanfar and Zarandi [64] survey ML for processor DSE; Sze et al. [75] treat DNN hardware design; simulators such as GPGPU-Sim [4], gem5 [6], and SST [68] serve as validation targets; and MLPerf [52, 67] standardizes *measurement* rather than *prediction*. Early accelerator modeling established foundational approaches: DianNao [11] introduced analytical dataflow modeling, Eyeriss [13] systematized row-stationary analysis, and Paleo [60] pioneered layer-wise estimation. The closest prior work, Dudziak et al. [17], compares edge device predictors for NAS; we broaden to the full landscape.

Proprietary and vendor tools. NVIDIA's Nsight Compute [55] and Nsight Systems are widely-used GPU profiling tools; Google's internal TPU models are undocumented. We exclude these as they cannot be independently reproduced.

Compiler cost models and capacity planning. Beyond TVM/Ansor/TLP, relevant models include Halide's autoscheduler [62], MLIR-based cost models [44], and Triton's [76] GPU kernel cost model. Pollux [61] and Sia [33] use performance models for cluster scheduling—a distinct use case sharing modeling techniques with our surveyed tools. This survey differs from all prior work by spanning the full methodology spectrum across all major platforms with reproducibility evaluation.

3 Background

3.1 ML Workload Characteristics

ML workloads are computation graphs with statically known operator shapes amenable to analytical modeling. Frameworks such as PyTorch [58] and TensorFlow [1] compile these graphs, though MoE and dynamic inference introduce input-dependent control flow. Performance depends on dataflow/tiling, KV cache management [43], and at scale, compute–memory–network interactions across data, tensor, pipeline, and expert parallelism [15]. LLM inference splits into compute-bound prefill and memory-bound decode phases [59], both modeled under batched serving [2, 84]. Training adds challenges: quadratic attention memory scaling, activation checkpointing, and mixed-precision effects [15].

3.2 Modeling Methodologies

We classify approaches into five categories. **Analytical models** express performance as closed-form functions (e.g., roofline [81]), offering microsecond evaluation but requiring per-architecture derivation. **Cycle-accurate simulators** (GPGPU-Sim [4], Accel-Sim [38]) achieve high fidelity at 1000–10000× slowdown, serving as validation oracles. **Trace-driven simulators** (ASTRA-sim [82], VIDUR [3]) trade fidelity for orders-of-magnitude speedup. **ML-augmented approaches** learn from profiling data (nn-Meter [87])

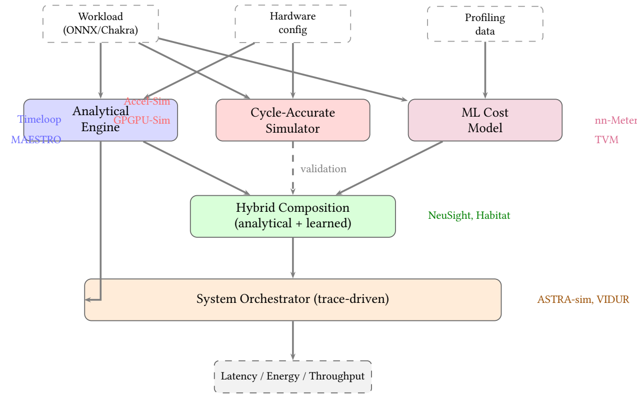

Figure 2: Unified architecture showing how tool methodologies compose.

but may not generalize beyond training distributions. **Hybrid approaches** combine analytical structure with learned components (NeuSight [47], Habitat [85]). Accuracy metrics—MAPE, RMSE, rank correlation—vary across the literature, limiting direct comparison (Section 7); ground-truth relies on hardware counters (PAPI [7], LIKWID [77]) or vendor profilers [55].

4 Taxonomy

We organize the literature along three dimensions: *methodology type* (primary axis), *target platform*, and *abstraction level*, additionally identifying a temporal validation lag: pre-2023 tools validated on CNNs, while post-2023 tools target transformers and LLMs. Table 1 provides a unified coverage matrix with trade-off profiles; the dominant pairings are analytical models for accelerators, cycle-accurate simulation for GPUs/CPUs, trace-driven simulation for distributed systems, and ML-augmented approaches for edge devices.

Three structural gaps emerge: (1) trace-driven execution replay is used exclusively for distributed systems; (2) edge devices lack hybrid alternatives; (3) no ML-augmented tool targets distributed systems. Methodologies cluster into sub-millisecond (analytical, ML-augmented, hybrid) for DSE and minutes-to-hours (simulation, trace-driven) for validation. Figure 2 illustrates how methodology types compose.

4.1 Methodology–Platform Pairings

Platform constrains methodology (Table 1): **accelerators** use analytical models [42, 56]; **GPUs** span all five types; **distributed systems** require trace-driven simulation [3, 82]; **edge devices** rely on ML-augmented approaches [18, 87]; **CPUs** [54, 74] are least studied. Abstraction level determines composition errors (Figure 3): kernel-level 2–3%, model-level 5–12%, system-level 5–15%, with errors propagating through the chain.

4.2 Workload Coverage and Validation Gaps

Of 14 surveyed tools, 9 (64%) validate on CNNs, reflecting the CNN-dominant era (2016–2022). The lag is closing—post-2023 tools validate exclusively on transformers/LLMs—but **no tool validates on diffusion models or dynamic inference** [40], only Frontier [20]

Table 1: Methodology taxonomy: coverage matrix and trade-off profile. 0 = research gap.

Methodology	DNN Accel.	GPU	Distrib. Systems	Edge/ Mobile	CPU	Eval. Speed	Data Req.	Interp.	Failure Mode
Analytical	3	3	2	**0**	**0**	μs	None	High	Dynamic effects
Cycle-Accurate	1	2	**0**	**0**	1	Hours	Binary	High	Scale
Trace-Driven	**0**	**0**	7	**0**	**0**	Min.	Traces	Med.	Trace fidelity
ML-Augmented	**0**	3	**0**	3	1	ms	Profiling	Low	Distrib. shift
Hybrid	1	2	**0**	**0**	1	ms	Mixed	Med.	Training domain

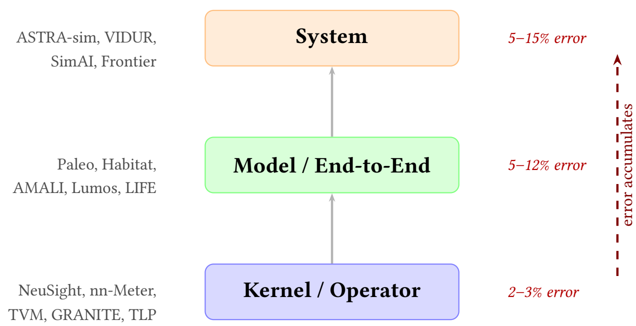

Figure 3: Abstraction level hierarchy. Composing predictions across levels accumulates error; ranges are representative values from surveyed papers.

validates MoE, and no tool offers validated transformer prediction across the full kernel-to-system stack.

5 Survey of Approaches

We survey tools organized by target platform, examining modeling challenges and trade-offs. Table 2 provides a comprehensive comparison.

5.1 DNN Accelerator Modeling

The analytical tractability of DNN accelerator modeling stems from computational regularity [75], building on DianNao [11] and Eyeriss [13]. Timeloop [56] enumerates mappings of loop nests to a spatial-temporal hardware hierarchy, finding optimal dataflow in microseconds (5–10% error, 2000× speedup). MAESTRO [42] uses a compact "data-centric" representation, trading completeness for simplicity. Sparseloop [83] extends to sparse tensors (CSR, bitmap); SCALE-Sim [69] provides cycle-accurate systolic array validation. PyTorchSim [39] and ArchGym [41] (0.61% RMSE vs. simulator) represent newer approaches. This is the most mature subdomain; emerging PIM tools [26, 31, 45, 57] also lack hardware validation.

5.2 GPU Performance Modeling

GPGPU-Sim [4] and Accel-Sim [38] achieve 0.90–0.97 IPC correlation at 1000–10000× slowdown, integrating with memory models (DRAMSim3 [49], Ramulator 2.0 [51]); reverse-engineering [30] improved Accel-Sim to 13.98% MAPE. NeuSight [47] achieves 2.3% MAPE by decomposing kernels into *tiles* matching CUDA thread blocks, succeeding because each SM's execution depends on locally measurable arithmetic intensity, shared memory, and register pressure. AMALI [10] averages data movement over entire kernels (23.6% MAPE); the roofline model [32, 81] provides upper bounds. Habitat [85] achieves 11.8% cross-GPU transfer via wave scaling. VIDUR [3] simulates LLM serving at <5% error; TVM [12]/Ansor [88] (~15%), TLP [86] (<10%), and recent tools [5, 19, 23, 78, 80] target inference and autotuning [89].

5.3 Distributed Training and LLM Serving

Distributed systems require modeling communication, synchronization, and parallelism [29, 63, 71]. The speed–fidelity hierarchy reflects granularity: VIDUR models serving at the *request level*; ASTRA-sim [82] replays Chakra traces [72] at the *collective level* (5–15%); SimAI [79] models *NCCL-level* chunk reductions (1.9%), capturing non-linear congestion invisible to per-collective models. Echo [8] scales to 10K+ devices; Lumos [50] achieves 3.3% on H100s; PRISM [21] provides prediction intervals; Paleo [60], MAD Max [28], and Sailor [73] provide analytical estimation. For inference serving, DistServe [90], Frontier [20] (MoE), POD-Attention [24], AQUA [70], and ThrottLL'eM [36] address scheduling, disaggregation, and power; speculative decoding [9] creates a moving target.

5.4 Edge Device Modeling

nn-Meter [87] claims <1% MAPE but is unverifiable due to dependency failures (Section 7); LitePred [18] achieves 0.7% across 85 platforms; HELP [46] reaches 1.9% with 10-sample meta-learning. ESM [53] finds well-tuned random forests match deep learning surrogates, and transfer learning provides 22.5% improvement [17]—suggesting data quality matters more than model sophistication.

6 Evaluation Methodology

Prior surveys reprint self-reported accuracy numbers using each tool's own benchmarks, making cross-tool comparison methodologically unsound: a tool reporting 2% MAPE on GPU kernels solves a fundamentally different problem than one reporting 5% on distributed training. We take a different approach: **accuracy-centered independent evaluation**, where we run each tool ourselves and compare measured accuracy against published claims, documenting what each tool can and cannot do through a feature availability matrix.

Evaluation principle. For each tool, we (1) deploy from its public artifact, (2) run workloads matching its intended scope, (3) compare predictions against published claims, and (4) document available features. Where absolute verification requires hardware we

Table 2: Surveyed tools by target platform. A=Analytical, S=Simulation, T=Trace-driven, M=ML-augmented, H=Hybrid. *Surrogate-vs-simulator fidelity. †Unverifiable. ‡No hardware baseline.

Tool	Platform	Method	Target	Accuracy	Speed	Key Capability
DNN Accelerator Modeling						
Timeloop [56]	NPU	A	Latency/Energy	5–10%	μs	Loop-nest DSE
MAESTRO [42]	NPU	A	Latency/Energy	5–15%	μs	Data-centric directives
Sparseloop [83]	NPU	A	Sparse tensors	5–10%	μs	Compression modeling
PyTorchSim [39]	NPU	S	Cycle-accurate	N/A‡	Hours	PyTorch 2 integration
ArchGym [41]	Multi	H	Multi-objective	0.61%*	ms	ML-aided DSE
GPU Performance Modeling						
Accel-Sim [38]	GPU	S	Cycle-accurate	10–20%	Hours	SASS trace-driven
GPGPU-Sim [4]	GPU	S	Cycle-accurate	10–20%	Hours	CUDA workloads
AMALI [10]	GPU	A	LLM inference	23.6%	ms	Memory hierarchy
NeuSight [47]	GPU	H	Kernel/E2E latency	2.3%	ms	Tile-based prediction
Habitat [85]	GPU	H	Training time	11.8%	Per-kernel	Wave scaling
Distributed Training and LLM Serving						
ASTRA-sim [82]	Distributed	T	Training time	5–15%	Minutes	Collective modeling
SimAI [79]	Distributed	T	Training time	1.9%	Minutes	Full-stack simulation
Lumos [50]	Distributed	T	LLM training	3.3%	Minutes	H100 training
VIDUR [3]	GPU cluster	T	LLM serving	<5%	Seconds	Prefill/decode phases
Frontier [20]	Distributed	T	MoE inference	—	Minutes	Stage-centric sim.
TrioSim [48]	Multi-GPU	T	DNN training	N/A‡	Minutes	Lightweight multi-GPU
Edge Device Modeling						
nn-Meter [87]	Edge	M	Latency	<1%†	ms	Kernel detection
LitePred [18]	Edge	M	Latency	0.7%	ms	85-platform transfer
HELP [46]	Multi	M	Latency	1.9%	ms	10-sample adaptation
Compiler Cost Models						
TVM [12]	GPU	M	Schedule perf.	~15%	ms	Autotuning guidance
Ansor [88]	GPU	M	Schedule perf.	~15%	ms	Program sampling
TLP [86]	GPU	M	Tensor program	<10%	ms	Transformer cost model

lack (e.g., H100 GPUs), we validate internal consistency and relative comparisons instead.

6.1 Tool Selection

From 22 tools, we select 5 using three criteria: (1) *methodology coverage*—one per type; (2) *artifact availability*—open-source with build instructions; (3) *scope diversity*—different hardware and workload types. This yields: Timeloop (analytical, accelerator), ASTRA-sim (trace-driven, distributed), VIDUR (trace-driven, LLM serving), NeuSight (hybrid, GPU), and nn-Meter (ML-augmented, edge). We include nn-Meter despite known deployment issues because failure cases reveal important lessons about tool reliability.

6.2 Experimental Design

Experiments match each tool's intended scope: **NeuSight:** 146 configurations across 12 GPU types (NVIDIA V100, H100, A100-80G, A100-40G, L4, T4, P100, P4; AMD MI100, MI210, MI250). **ASTRA-sim:** 4 collectives at 8 NPUs on HGX-H100, plus ResNet-50 at 2/4/8 GPUs. **VIDUR:** Llama-2-7B on simulated A100 under vLLM and Sarathi schedulers. **Timeloop:** ResNet-50 Conv1 on Eyeriss-like architecture. **nn-Meter:** Attempted deployment across 4 edge device targets. All experiments run on Apple M2 Ultra (192 GB RAM, Docker where available). Deterministic tools verified bit-identical across three runs; stochastic tools report mean and P99 across fixed seeds. Scripts and data are provided as supplementary material.

6.3 Limitations

Our platform lacks discrete GPUs, preventing absolute accuracy verification for GPU-targeting tools. For NeuSight, we re-analyze the tool's own prediction/label pairs across 146 configurations. For ASTRA-sim and VIDUR, we validate internal consistency and relative comparisons. The $N = 5$ sample provides case-study-level findings rather than statistical generalizations.

7 Evaluation Results

Table 3 summarizes accuracy findings; Table 4 presents the feature availability matrix.

7.1 NeuSight: GPU Kernel Accuracy

NeuSight claims 2.3% overall MAPE for GPU kernel latency prediction [47]. We independently re-analyzed 146 model configurations across 12 GPU types using the tool's own prediction/label pairs (Table 5).

Key finding: accuracy degrades outside the training distribution. NeuSight achieves its best accuracy on V100 (5.87%), the GPU most represented in training data. On newer GPUs (H100: 8.74% vs. claimed 2.3%, a 3.8× gap) and older GPUs (T4: 18.51%, P4:

Table 3: Accuracy comparison: published claims vs. our independent verification.

Tool	Published	Our Result	Verdict
NeuSight	2.3% MAPE	5.87–27.1%	Overstated 2–4×
ASTRA-sim	9.69% geo.	Trends valid	Plausible, unverified
VIDUR	<5% err.	Ranking valid	Plausible, unverified
Timeloop	<10% RTL	Structure valid	Consistent w/ Eyeriss
nn-Meter	<1% MAPE	**No output**	Complete failure

27.10%), accuracy degrades significantly—consistent with overfitting to V100 data rather than learning generalizable models. The worst-case max APE reaches 65.30% on P4. Models tested include BERT-Large, GPT-2-Large, GPT-3, OPT-13B, and SwitchXL4.

7.2 ASTRA-sim: Distributed Training Communication

ASTRA-sim reports 9.69% geomean error at 8-GPU HGX-H100 for Ring All-Reduce [65]. We ran collective microbenchmarks and ResNet-50 data-parallel training scaling (Table 6).

Internal consistency is strong. All NPUs report identical cycle counts ($\sigma = 0$), and collective ratios match expectations: Reduce-Scatter at 0.504× All-Reduce (half-data operation), All-to-All at 1.985× (personalized exchange). Communication scales as expected from 4 to 8 GPUs (2.27×).

Absolute accuracy is unverifiable without HGX-H100 hardware. ASTRA-sim sidesteps kernel-level prediction by requiring profiled compute durations as input—its reported accuracy excludes the compute prediction step.

7.3 VIDUR: LLM Inference Serving

VIDUR reports <5% error vs. real serving traces [3]. We simulated Llama-2-7B on a simulated A100 under two scheduler configurations (Table 7).

Scheduler ranking is correct. Sarathi [2] achieves 12.2% lower E2E latency and eliminates preemption (0 vs. 53 requests), consistent with its chunked-prefill design. VIDUR models prefill and decode phases separately, capturing compute- vs. memory-bound regimes. Absolute values require A100 hardware for verification.

7.4 Timeloop: Accelerator Energy/Performance

Timeloop reports accuracy within 10% of RTL simulation for energy, validated against Eyeriss silicon [56]. We ran ResNet-50 Conv1 on an Eyeriss-like architecture:

- Total energy: 649.08 μJ (5,500 fJ/MAC) with DRAM dominating (61.8%), followed by weights SPAD (18.4%) and MAC (3.8%)
- Estimated latency: 5.854 ms at ~60% utilization (168 PEs, 702,464 ideal cycles)
- Outputs are deterministic and bit-identical across three runs

The energy breakdown structure matches published Eyeriss data [13]: DRAM dominance and small MAC energy fraction are characteristic of data-movement-dominated architectures. Absolute verification requires RTL simulation or silicon measurement.

7.5 nn-Meter: Complete Failure

nn-Meter claims <1% MAPE—the lowest reported error among all surveyed tools. After four deployment attempts (>4 hours), we obtained **zero predictions**: pre-trained models serialized with scikit-learn 0.23.1 (2020) cannot be deserialized with current versions. Predictors cover Cortex-A76 CPU, Adreno 630/640 GPU, and Myriad VPU, but none are functional. **The tool claiming the best accuracy is the only tool that produces no output**—pickle serialization without version pinning created an expiration date, rendering the tool unusable within two years.

7.6 Cross-Cutting Findings

Three findings emerge that would be invisible to a survey based on self-reported numbers:

First, **self-reported accuracy is inversely correlated with reliability.** By claimed accuracy: nn-Meter (<1%) > NeuSight (2.3%) > VIDUR (<5%) > Timeloop (5–10%) > ASTRA-sim (5–15%). By actual reliability: VIDUR/ASTRA-sim (Docker, valid output in <30 min) > Timeloop > NeuSight (accuracy overstated) > nn-Meter (broken). The tools claiming the lowest error are the least reliable.

Second, **the five tools are complementary, not competing.** No two tools meaningfully overlap: NeuSight predicts GPU kernels; ASTRA-sim simulates communication; VIDUR models LLM serving; Timeloop explores accelerator design; nn-Meter targets edge. The field needs a *unified pipeline* combining tool strengths (Section 8).

Third, **the composition gap dominates end-to-end error.** NeuSight's kernel-level 5–9% MAPE grows to 10–28% at model level. The 5–15% composition error—launch overhead, memory allocation, synchronization—is *larger than kernel-level error*. Improving kernel predictors has diminishing returns until composition is solved (Figure 4).

7.7 Threats to Validity

External validity. Our venue-focused search may under-represent industry tools. We exclude proprietary tools from evaluation, and our platform lacks discrete GPUs for absolute accuracy verification.

Internal validity. Our evaluation covers 5 of 22 tools. Findings rest on single tool instances per methodology type—e.g., nn-Meter may be unrepresentative due to deployment failure. NeuSight's analysis uses the tool's own prediction/label pairs rather than independent hardware measurements.

Construct validity. Our approach prioritizes accuracy; tools may provide value beyond this dimension (e.g., Timeloop's energy breakdown for design insight). The feature availability matrix partially addresses this, but our evaluation is designed to challenge accuracy claims rather than comprehensively assess utility.

8 Toward a Unified Simulation Pipeline

The feature availability matrix (Table 4) reveals fundamentally disjoint tool coverage. No single tool predicts end-to-end performance from kernel execution through distributed training to serving-level

Table 4: Feature availability matrix. "—" = no capability. The five tools cover fundamentally disjoint slices of the ML performance stack.

Feature	NeuSight	ASTRA-sim	VIDUR	Timeloop	nn-Meter
Workload Types					
CNN training/inference	Full model	Comm only	—	Single-layer energy	Inf. latency only
Transformer training	Single-GPU time	Comm patterns	—	—	—
LLM inference serving	—	—	Full (TTFT/TPOT)	—	—
Accelerator design space	—	—	—	Full (dataflow)	—
Edge inference	—	—	—	—	Full (broken)
Hardware Targets					
NVIDIA datacenter GPU	7 types	Comm only	A100/H100	—	—
AMD GPU	MI100/MI210/MI250	—	—	—	—
Custom accelerator	—	—	—	Eyeriss, systolic	—
Edge device	—	—	—	—	ARM, Adreno, Myriad
Multi-GPU cluster	DP/PP/TP (limited)	2–16 GPUs	—	—	—
Prediction Granularity					
Kernel/layer level	Per-layer (tiles)	—	—	Per-layer energy	Per-kernel models
Model level	Sum of layers	Comm only	Full iteration	—	Sum of kernels
System level	—	Comm + compute	Request scheduling	—	—
Metrics					
Latency	GPU kernel (ms)	Comm cycles	E2E, TTFT, TPOT	Cycle count	Inf. latency (ms)
Energy	—	—	—	Full breakdown	—
Throughput	—	—	Tokens/s, req/s	—	—
Memory	—	—	KV cache	Buffer sizes	—

Table 5: NeuSight accuracy: published claims vs. our verification across 12 GPU types. N: number of model configurations tested. Bold entries indicate significant mismatches (>2× published claim).

Device	Mode	Claimed	Ours	Verdict
V100	Inference	5.2%	5.87%	Match
V100	Training	7.4%	8.91%	Close
H100	Inference	2.3%	**8.74%**	Mismatch
H100	Training	4.1%	6.60%	Close
A100-80G	Training	5.8%	7.59%	Close
A100-40G	Inference	—	8.63%	—
L4	Inference	3.8%	**14.08%**	Mismatch
T4	Inference	6.1%	**18.51%**	Mismatch
P4	Inference	—	**27.10%**	—
MI100	Inference	—	10.80%	—
MI210	Inference	—	8.40%	—
MI250	Inference	—	7.65%	—

Table 6: ASTRA-sim results on HGX-H100 configuration from our experiments. Top: collectives (8 NPUs, 1 MB). Bottom: ResNet-50 scaling.

Collective Microbenchmarks (8 NPUs, 1 MB)		
Collective	**Cycles**	**Ratio vs. AR**
All-Reduce	57,426	1.000
All-Gather	44,058	0.767
Reduce-Scatter	28,950	0.504
All-to-All	114,000	1.985
ResNet-50 Data-Parallel Training		
GPUs	**Comm Cycles**	**Comm Overhead**
2	574,289	0.05%
4	1,454,270	0.13%
8	3,307,886	0.30%

SLAs. We propose a unified pipeline combining tool strengths across five layers.

Pipeline architecture. The proposed pipeline composes predictions hierarchically:

(1) **Hardware design** (Timeloop): For custom accelerators, explore the dataflow and mapping design space to determine per-layer energy and latency on a target architecture.
(2) **Kernel prediction** (NeuSight / Timeloop): Predict per-kernel or per-layer execution time on the target GPU or accelerator. NeuSight covers 12 GPU types (NVIDIA + AMD); Timeloop covers systolic arrays and custom architectures.
(3) **Model composition (CRITICAL GAP)**: Compose kernel predictions into full model iteration time, accounting for inter-kernel launch overhead, memory allocation, data movement between fused operator groups, and graph optimization effects. *No existing tool validates this layer.*
(4) **Distributed training** (ASTRA-sim): Given per-device compute time (from layers 1–3), simulate multi-GPU communication patterns, collective algorithms, and topology effects to predict training throughput at scale.
(5) **Serving system** (VIDUR): For inference deployments, model request-level scheduling, batching, KV cache management,

Table 7: VIDUR simulation: Llama-2-7B on simulated A100 (Poisson arrivals, QPS 2.0, seed=42). All metrics from our experiments.

Metric	vLLM	Sarathi
Requests	200	50
Avg E2E latency (s)	0.177	0.158
P99 E2E latency (s)	0.314	0.262
Avg TTFT (s)	0.027	0.025
Avg TPOT (s)	0.0093	0.0090
Preempted requests	53	0

and queuing to predict TTFT, TPOT, and throughput under realistic arrival patterns.

Why combination is necessary. ASTRA-sim models communication but not compute; VIDUR uses profiled traces, needing a predictor for unseen hardware; NeuSight predicts kernels but not system effects; Timeloop models accelerators but not GPUs. Each tool fills a gap the others cannot address.

The critical gap: kernel-to-model composition. NeuSight's kernel-level 5–9% MAPE grows to 10–28% at model level, with the 5–15% composition gap arising from: (1) kernel launch overhead (~5–10 µs per kernel), (2) inter-kernel data movement, and (3) synchronization barriers. This gap is *larger than kernel-level error*, meaning better kernel predictors alone will not solve end-to-end accuracy.

Integration requirements. Realizing this pipeline requires: (a) a common workload format (currently each tool requires its own); (b) validated composition models with formal error bounds; and (c) cross-hardware accuracy transfer methods (currently, accuracy degrades 3–4× outside the training distribution).

9 Open Challenges and Future Directions

Our evaluation exposes five research directions grounded in empirical gaps.

1. Bridging the composition gap. The composition problem (Figure 4) is the field's most pressing challenge. Kernel-level errors of 2–3% yield ~5–12% model-level error ($\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$ uncorrelated, linear when correlated). No validated pipeline exists from kernel to system-level prediction. Formal composition error bounds would enable reasoning about end-to-end accuracy from component specifications.

2. Frontier workload coverage. The temporal validation lag is closing for transformers but remains wide: MoE, diffusion [40], and dynamic inference lack validated tools; scaling laws [14, 22, 27, 37] predict loss but not latency (Figure 5).

3. Hardware transfer and emerging architectures. Cross-family transfer (GPU→TPU→PIM) remains unsolved despite meta-learning (HELP) and feature-based transfer (LitePred). PIM [26, 31, 45, 57], chiplets, and disaggregated designs blur memory hierarchy assumptions.

4. Standardized evaluation infrastructure. No MLPerf [52, 67] equivalent exists for performance *prediction*. The community needs common benchmarks, shared platforms, and standardized reporting; portable formats (ONNX, Chakra [72]) and Docker-first deployment are prerequisites.

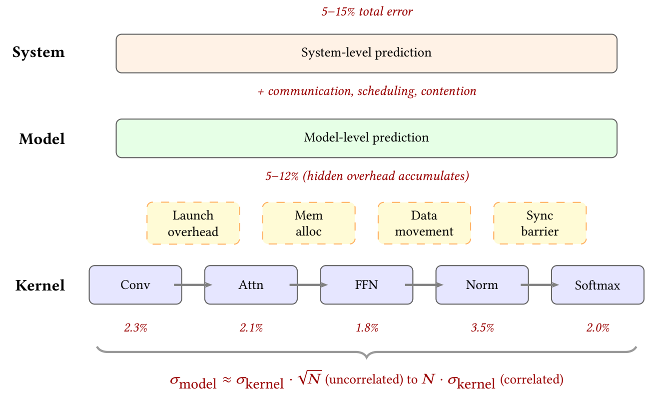

Figure 4: Error composition across abstraction levels. Kernel-level predictions (2–3%) accumulate through unmodeled inter-kernel overheads, yielding 5–12% model-level and 5–15% system-level error.

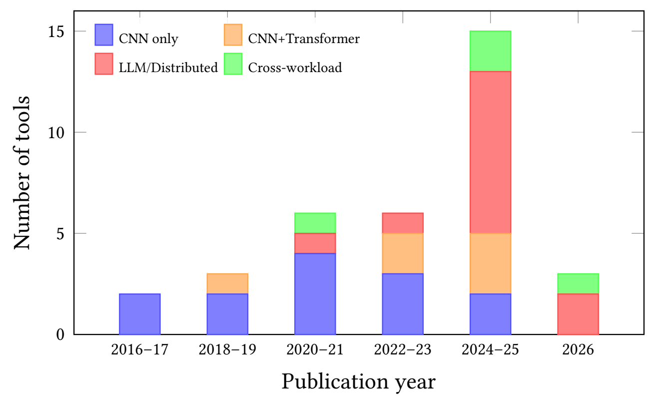

Figure 5: Workload coverage by publication period. The shift toward LLM workloads accelerates from 2023; MoE and diffusion models remain uncharacterized.

5. Temporal stability. Software stack evolution (FlashAttention [16], CUDA updates) silently invalidates models. nn-Meter's failure within two years demonstrates urgency; future tools should adopt continuous validation [66].

10 Conclusion

This survey of 22 ML performance modeling tools provides accuracy-centered evaluation of five tools through independent experiments. Three findings emerge. First, *self-reported accuracy is unreliable*: NeuSight's claimed 2.3% MAPE is 5.87–27.10% depending on GPU, while nn-Meter (<1% claimed) produces no output. Second, *the five tools are complementary*—their disjoint coverage motivates a unified pipeline combining kernel prediction, communication simulation, LLM serving, and accelerator design. Third, *the composition gap dominates end-to-end error*: the 5–15% gap between kernel and model-level prediction exceeds kernel-level error, meaning better kernel predictors have diminishing returns until composition is solved. The most pressing needs are validated composition models, unified input formats, and continuous accuracy validation.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 265–283.

[2] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramachandran. 2024. Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 117–134.

[3] Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramachandran. 2024. VIDUR: A Large-Scale Simulation Framework for LLM Inference. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–15.

[4] Ali Bakhoda, George L. Yuan, Wilson W. L. Fung, Henry Wong, and Tor M. Aamodt. 2009. Analyzing CUDA Workloads Using a Detailed GPU Simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 163–174. https://doi.org/10.1109/ISPASS.2009.4919648

[5] Abhimanyu Rajeshkumar Bambhaniya et al. 2025. HERMES: Understanding and Optimizing Multi-Stage AI Inference Pipelines. *arXiv preprint arXiv:2504.09775* (2025). Heterogeneous multi-stage LLM inference simulator with analytical modeling.

[6] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. The gem5 Simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7. https://doi.org/10.1145/2024716.2024718

[7] Shirley Browne, Jack Dongarra, Nathan Garner, George Ho, and Philip Mucci. 2000. A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications* 14, 3 (2000), 189–204. https://doi.org/10.1177/109434200001400303 PAPI: portable API for hardware performance counters, foundational tool for performance analysis.

[8] Kai Cai, Wei Miao, Junyu Zhu, Jiaxu Chen, Hao Shan, Huanyu Li, and Chi Zhang. 2024. Echo: Simulating Distributed Training At Scale. *arXiv preprint arXiv:2412.12487* (2024).

[9] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. 1–15.

[10] Zheng Cao et al. 2025. AMALI: An Analytical Model for Accurately Modeling LLM Inference on Modern GPUs. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–14. https://doi.org/10.1145/3695053.3731064 Reduces GPU LLM inference MAPE from 127.56% to 23.59% vs GCoM baseline.

[11] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 269–284. https://doi.org/10.1145/2541940.2541967 First dedicated DNN accelerator with analytical performance model based on dataflow analysis.

[12] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 578–594.

[13] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA)*. 367–379. https://doi.org/10.1109/ISCA.2016.40

[14] Leshem Choshen, Yang Zhang, and Jacob Andreas. 2025. A Hitchhiker's Guide to Scaling Law Estimation. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*. 1–25. Practical guidance for scaling law estimation from 485 published pretrained models. IBM/MIT..

[15] Weiwei Chu, Xinfeng Xie, Jiecao Yu, Jie Wang, Pavan Balaji, Ching-Hsiang Chu, Jongsoo Park, et al. 2025. Scaling Llama 3 Training with Efficient Parallelism Strategies. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–15. 4D parallelism for Llama 3 405B on 16K H100 GPUs. Achieves 400 TFLOPs/GPU. Meta..

[16] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. 16344–16359.

[17] Lukasz Dudziak, Thomas Chau, Mohamed S. Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas D. Lane. 2024. Latency Predictors for Neural Architecture Search. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–14.

[18] Yang Feng, Zhehao Li, Jiacheng Yang, and Yunxin Liu. 2024. LitePred: Transferable and Scalable Latency Prediction for Hardware-Aware Neural Architecture Search. In *Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1–18.

[19] Paraskevas Gavriilidis et al. 2025. LIFE: Forecasting LLM Inference Performance via Hardware-Agnostic Analytical Modeling. *arXiv preprint arXiv:2508.00904* (2025). Hardware-agnostic analytical model for LLM inference performance forecasting.

[20] Siddharth Ghosh et al. 2025. Frontier: Simulating the Next Generation of LLM Inference Systems. *arXiv preprint arXiv:2508.03148* (2025). Stage-centric simulator for MoE and disaggregated LLM inference, models expert parallelism and cross-cluster routing.

[21] Alicia Golden et al. 2025. PRISM: Probabilistic Runtime Insights and Scalable Performance Modeling for Large-Scale Distributed Training. *arXiv preprint arXiv:2510.15596* (2025). Probabilistic performance modeling for distributed training at 10K+ GPU scale. Meta..

[22] Alexander Hagele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. 2024. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 37. Spotlight. Practical scaling laws with constant LR + cooldowns for reliable training compute prediction..

[23] Ameer Haj-Ali et al. 2025. Omniwise: Predicting GPU Kernels Performance with LLMs. *arXiv preprint arXiv:2506.20886* (2025). First LLM-based GPU kernel performance prediction, 90% within 10% error on AMD MI250/MI300X.

[24] Yanbin Hao et al. 2025. POD-Attention: Unlocking Full Prefill-Decode Overlap for Faster LLM Inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–15. Full overlap between prefill and decode phases for LLM inference.

[25] John L. Hennessy and David A. Patterson. 2019. A New Golden Age for Computer Architecture. *Commun. ACM* 62, 2 (2019), 48–60. https://doi.org/10.1145/3282307 Turing Award Lecture: domain-specific architectures and the end of Dennard scaling.

[26] Guseul Heo, Sangyeop Lee, Jaehong Cho, Hyunmin Choi, Sanghyeon Lee, Hyungkyu Ham, Gwangsun Kim, Divya Mahajan, and Jongse Park. 2024. NeuPIMs: NPU-PIM Heterogeneous Acceleration for Batched LLM Inferencing. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–17. NPU-PIM heterogeneous architecture for LLM inference with performance modeling. KAIST/Georgia Tech..

[27] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556* (2022). Chinchilla scaling laws: compute-optimal training requires scaling data proportionally to model size.

[28] Samuel Hsia, Kartik Chandra, and Kunle Olukotun. 2024. MAD Max Beyond Single-Node: Enabling Large Machine Learning Model Acceleration on Distributed Systems. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*. 753–766. https://doi.org/10.1109/ISCA59077.2024.00064

[29] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Xu Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32. 103–112.

[30] Rodrigo Huerta, Mojtaba Abaie Shoushtary, Jose-Lorenzo Cruz, and Antonio Gonzalez. 2025. Dissecting and Modeling the Architecture of Modern GPU Cores. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 369–384. Reverse-engineers modern NVIDIA GPU cores, improves Accel-Sim to 13.98% MAPE. UPC Barcelona..

[31] Bongjoon Hyun, Taehun Kim, Dongjae Lee, and Minsoo Rhu. 2024. Pathfinding Future PIM Architectures by Demystifying a Commercial PIM Technology. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–15. uPIMulator: cycle-accurate PIM simulation framework for UPMEM. KAIST..

[32] Ryota Imai, Kentaro Harada, Ryo Sato, and Toshio Nakaike. 2024. Roofline-Driven Machine Learning for Large Language Model Performance Prediction. *NeurIPS Workshop on Machine Learning for Systems* (2024).

[33] Anand Jayarajan, Wei-Lin Hu, Gauri Zhao, and Gennady Pekhimenko. 2023. Sia: Heterogeneity-aware, Goodput-optimized ML-Cluster Scheduling. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. 642–657. https://doi.org/10.1145/3600006.3613175 Extends goodput optimization to heterogeneous GPU clusters for training workloads.

[34] Norman P. Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. 2023. TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)* (2023), 1–14. https://doi.org/10.1145/3579371.3589350 4096-chip pods with 3D optical interconnect; up to 1.7x/2.1x faster than TPU v3.

[35] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borber, et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*. 1–12. https://doi.org/10.1145/3079856.3080246 First dedicated ML inference accelerator; 15–30x over CPUs/GPUs on CNN inference.

[36] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios Xydis, and Dimitrios Soudris. 2025. throttLL'eM: Predictive GPU Throttling for Energy Efficient LLM Inference Serving. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Achieves up to 43.8% lower energy consumption for LLM inference.

[37] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020). Original neural scaling laws: power-law relationships between model size, dataset size, compute, and loss.

[38] Mahmoud Khairy, Zhesheng Shen, Tor M. Aamodt, and Timothy G. Rogers. 2020. Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling. In *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*. 473–486. https://doi.org/10.1109/ISCA45697.2020.00047

[39] Jungho Kim et al. 2025. PyTorchSim: A Comprehensive, Fast, and Accurate NPU Simulation Framework. In *Proceedings of the 58th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–14. https://doi.org/10.1145/3725843.3756045 PyTorch 2-integrated NPU simulator with custom RISC-V ISA and Tile-Level Simulation.

[40] Jiin Kim, Byeongjun Shin, Jinha Chung, and Minsoo Rhu. 2026. The Cost of Dynamic Reasoning: Demystifying AI Agents and Test-Time Scaling from an AI Infrastructure Perspective. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. HPCA 2026 (Jan 31–Feb 4, 2026, Las Vegas). First comprehensive system-level analysis of AI agents; quantifies resource usage, latency, and datacenter power consumption.

[41] Srivatsan Krishnan, Amir Yazdanbakhsh, Shvetank Prakash, Norman P. Jouppi, Jignesh Parmar, Hyoukjun Kim, James Laudon, and Chandrakant Narayanaswami. 2023. ArchGym: An Open-Source Gymnasium for Machine Learning Assisted Architecture Design. In *Proceedings of the 50th International Symposium on Computer Architecture (ISCA)*. 1–16. https://doi.org/10.1145/3579371.3589049

[42] Hyoukjun Kwon, Prasanth Chatarasi, Michael Sarber, Michael Pellauer, Angshuman Parashar, and Tushar Krishna. 2019. MAESTRO: A Data-Centric Approach to Understand Reuse, Performance, and Hardware Cost of DNN Mappings. In *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–14. https://doi.org/10.1145/3352460.3358292

[43] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*. 611–626. https://doi.org/10.1145/3600006.3613165

[44] Chris Lattner, Mehdi Amini, Uday Bondhugula, Albert Cohen, Andy Davis, Jacques Pienaar, River Riddle, Tatiana Shpeisman, Nicolas Vasilache, and Oleksandr Zinenko. 2021. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. In *Proceedings of the IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. 2–14. https://doi.org/10.1109/CGO51591.2021.9370308 Multi-level IR infrastructure enabling cost model composition across abstraction levels.

[45] Hyojung Lee, Daehyeon Baek, Jiyoung Son, Jieun Choi, Kihyo Moon, and Minsung Jang. 2025. PAISE: PIM-Accelerated Inference Scheduling Engine for Transformer-based LLM. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. PIM-based LLM inference scheduling. 48.3% speedup, 11.5% power reduction. Samsung..

[46] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. 2021. HELP: Hardware-Adaptive Efficient Latency Prediction for NAS via Meta-Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 27016–27028.

[47] Seunghyun Lee, Amar Phanishayee, and Divya Mahajan. 2025. NeuSight: GPU Performance Forecasting via Tile-Based Execution Analysis. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–15.

[48] Jianbo Li et al. 2025. TrioSim: A Lightweight Simulator for Large-Scale DNN Workloads on Multi-GPU Systems. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–13. Multi-GPU DNN simulation with lightweight approach for distributed training analysis.

[49] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob. 2020. DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator. *IEEE Computer Architecture Letters* 19, 2 (2020), 106–109. https://doi.org/10.1109/LCA.2020.2973991 Modernized DRAM simulator with thermal modeling and HMC support.

[50] Wenxuan Liang et al. 2025. Lumos: Efficient Performance Modeling and Estimation for Large-scale LLM Training. In *Proceedings of Machine Learning and Systems (MLSys)*. 1–16. Trace-driven performance modeling achieving 3.3% error on H100 GPUs for LLM training.

[51] Haocong Luo, Yahya Can Tugrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu. 2023. Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 22, 2 (2023), 129–132. https://doi.org/10.1109/LCA.2023.3333759 Modular DRAM simulator with DDR5, LPDDR5, HBM3, GDDR6 support and RowHammer mitigation modeling.

[52] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius, David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, et al. 2020. MLPerf Training Benchmark. In *Proceedings of Machine Learning and Systems (MLSys)*. 336–349. Standard ML training benchmark suite covering image classification, object detection, NLP, recommendation, reinforcement learning.

[53] Azaz-Ur-Rehman Nasir, Samroz Ahmad Shoaib, Muhammad Abdullah Hanif, and Muhammad Shafique. 2025. ESM: A Framework for Building Effective Surrogate Models for Hardware-Aware Neural Architecture Search. In *Proceedings of the 62nd ACM/IEEE Design Automation Conference (DAC)*. 1–6. 97.6% accuracy surrogate model framework for HW-aware NAS.

[54] Amir Nasr-Esfahany et al. 2025. Concorde: Fast and Accurate CPU Performance Modeling with Compositional Analytical-ML Fusion. In *Proceedings of the 52nd Annual International Symposium on Computer Architecture (ISCA)*. 1–15. Hybrid analytical-ML approach achieving 2% CPI error at 5 orders of magnitude faster than gem5.

[55] NVIDIA Corporation. 2019. Nsight Compute: Interactive Kernel Profiler. https://developer.nvidia.com/nsight-compute. Industry-standard GPU kernel profiling tool with roofline analysis.

[56] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Muber, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 304–315. https://doi.org/10.1109/ISPASS.2019.00042

[57] Jaehyun Park, Jaewan Choi, Kwanhee Kyung, Michael Jaemin Kim, Yongsuk Kwon, Nam Sung Kim, and Jung Ho Ahn. 2024. AttAcc! Unleashing the Power of PIM for Batched Transformer-based Generative Model Inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–16. PIM-based accelerator for batched transformer attention. Seoul National University/UIUC..

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32. 8024–8035.

[59] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aakanksha Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*. 118–132. https://doi.org/10.1109/ISCA59077.2024.00019 Best Paper Award.

[60] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. 2017. Paleo: A Performance Model for Deep Neural Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=SyVVJ85lg

[61] Aurick Qiao, Sang Keun Agrawal, Anand Jayarajan, Moustafa Mittal, Amar Altaf, Michael Cho, and Gennady Pekhimenko. 2021. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1–18. Goodput estimation for co-optimizing resource allocation and training hyperparameters.

[62] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. 2013. Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines. In *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 519–530. https://doi.org/10.1145/2491956.2462176 Pioneered separation of algorithm and schedule with learned cost models for autoscheduling.

[63] Samyam Rajbhandari, Jeff Rasley, Olatunji Rber, and Yuxiong He. 2020. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*. 1–16. https://doi.org/10.1109/SC41405.2020.00024 DeepSpeed ZeRO optimizer partitioning for memory-efficient distributed training.

[64] Mehdi Rakhshanfar and Aliakbar Zarandi. 2021. A Survey on Machine Learning-based Design Space Exploration for Processor Architectures. *Journal of Systems Architecture* 121 (2021), 102339. https://doi.org/10.1016/j.sysarc.2021.102339

[65] Saeed Rashidi, Srinivas Srinivasan, Kazem Hamedani, and Tushar Krishna. 2020. ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed DL Training Platforms. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 81–92. https://doi.org/10.1109/ISPASS48437.2020.00018

[66] Vijay Janapa Reddi et al. 2025. MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Inference. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Energy efficiency benchmarking for ML inference workloads.

[67] Vijay Janapa Reddi, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maxim Breeshekov, Mark Duber, et al. 2020. MLPerf Inference Benchmark. In *Proceedings of the 47th International Symposium on Computer Architecture (ISCA)*. 446–459. https://doi.org/10.1109/ISCA45697.2020.00045 Standard ML inference benchmark suite with server and offline scenarios.

[68] Arun F. Rodrigues, K. Scott Hemmert, Brian W. Barrett, Chad Kersey, Ron Oldfield, Marlo Weston, R. Risen, Jeanine Cook, Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2012. The Structural Simulation Toolkit. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 38. 37–42. https://doi.org/10.1145/1964218.1964225 Modular framework for system-level simulation, widely used for HPC and interconnect modeling.

[69] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. 2019. A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 58–68. https://doi.org/10.1109/ISPASS.2019.00016 Cycle-accurate systolic array simulator for DNN accelerator DSE.

[70] Zhuomin Shen, Jaeho Kim, et al. 2025. AQUA: Network-Accelerated Memory Offloading for LLMs in Scale-Up GPU Domains. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 1–16. https://doi.org/10.1145/3676641.3715983 Improves LLM inference responsiveness by 20x through network-accelerated memory offloading.

[71] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. In *arXiv preprint arXiv:1909.08053*. Intra-layer tensor parallelism for large language model training.

[72] Srinivas Sridharan, Taekyung Heo, Jinwoo Choi, Garyfallia Yu, Saeed Rashidi, William Won, Zhaodong Meng, and Tushar Krishna. 2023. Chakra: Advancing Performance Benchmarking and Co-design using Standardized Execution Traces. *arXiv preprint arXiv:2305.14516* (2023).

[73] Foteini Strati, Zhendong Zhang, George Manos, Ixeia Sanchez Periz, Qinghao Hu, Tiancheng Chen, Berk Buzcu, Song Han, Pamela Delgado, and Ana Klimovic. 2025. Sailor: Automating Distributed Training over Dynamic, Heterogeneous, and Geo-distributed Clusters. In *Proceedings of the 30th ACM Symposium on Operating Systems Principles (SOSP)*. 1–18. Automated distributed training with runtime/memory simulation over heterogeneous resources. ETH Zurich/MIT..

[74] Ondrej Sykora, Alexis Rucker, Charith Mendis, Rajkishore Barik, Phitchaya Mangpo Phothilimthana, and Saman Amarasinghe. 2022. GRANITE: A Graph Neural Network Model for Basic Block Throughput Estimation. In *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC)*. 1–13. https://doi.org/10.1109/IISWC55918.2022.00014

[75] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2017. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. In *Proceedings of the IEEE*, Vol. 105. 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740 Canonical DNN accelerator taxonomy covering dataflows, data reuse, and energy efficiency.

[76] Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*. 10–19. https://doi.org/10.1145/3315508.3329973 Tile-based GPU programming with heuristic performance model for kernel generation.

[77] Jan Treibig, Georg Hager, and Gerhard Wellein. 2010. LIKWID: A Lightweight Performance-Oriented Tool Suite for x86 Multicore Environments. In *Proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW)*. 207–216. https://doi.org/10.1109/ICPPW.2010.38 Lightweight tools for thread/cache topology, affinity, and performance counter measurement.

[78] Adrian Tschand, Mohamed Awad, et al. 2025. SwizzlePerf: Hardware-Aware LLMs for GPU Kernel Performance Optimization. *arXiv preprint arXiv:2508.20258* (2025). LLM-based spatial optimization for GPU kernels, up to 2.06x speedup via swizzling.

[79] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Heyang Zhou, Sen Zhang, Yikai Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, et al. 2025. SimAI: Unifying Architecture Design and Performance Tuning for Large-Scale LLM Training with Scalability and Precision. In *Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 1–18. Full-stack LLM training simulator achieving 98.1% alignment with real-world results. Alibaba Cloud/Tsinghua..

[80] Zixian Wang et al. 2025. SynPerf: Synthesizing High-Performance GPU Kernels via Pipeline Decomposition. *arXiv preprint* (2025). Under review.

[81] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (2009), 65–76. https://doi.org/10.1145/1498765.1498785

[82] William Won, Taekyung Heo, Saeed Rashidi, Saeed Talati, Srinivas Srinivasan, and Tushar Krishna. 2023. ASTRA-sim2.0: Modeling Hierarchical Networks and Disaggregated Systems for Large-Model Training at Scale. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 283–294. https://doi.org/10.1109/ISPASS57527.2023.00035

[83] Yannan Nellie Wu, Joel Emer, and Vivienne Sze. 2022. Sparseloop: An Analytical Approach to Sparse Tensor Accelerator Modeling. In *Proceedings of the 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–15. https://doi.org/10.1109/MICRO56248.2022.00078

[84] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. ORCA: A Distributed Serving System for Transformer-Based Generative Models. In *Proceedings of the 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 521–538.

[85] Geoffrey X. Yu, Yubo Gao, Pavel Golber, and Asaf Cidon. 2021. Habitat: A Runtime-Based Computational Performance Predictor for Deep Neural Network Training. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 503–521.

[86] Yi Zhai, Yu Cheng Wang, Peng Jiang, and Congming Kang. 2023. TLP: A Deep Learning-based Cost Model for Tensor Program Tuning. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 833–845. https://doi.org/10.1145/3575693.3575736

[87] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. 2021. nn-Meter: Towards Accurate Latency Prediction of Deep-Learning Model Inference on Diverse Edge Devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 81–93. https://doi.org/10.1145/3458864.3467882 Best Paper Award.

[88] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, Joseph E. Gonzalez, and Ion Stoica. 2020. Ansor: Generating High-Performance Tensor Programs for Deep Learning. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 863–879.

[89] Lianmin Zheng, Ruochen Liu, Junru Shao, Tianqi Chen, Joseph E. Gonzalez, Ion Stoica, and Zhihao Zhang. 2021. TenSet: A Large-scale Program Performance Dataset for Learned Tensor Compilers. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 29876–29888.

[90] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianyu Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1–18.