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

As machine learning workloads grow in scale and complexity—spanning training and inference for CNNs, transformers, mixture-of-experts models, and LLMs—architects and system designers need fast, accurate methods to predict their performance across diverse hardware platforms. This survey provides a comprehensive analysis of the tools and methods available for modeling and simulating the performance of ML workloads, covering analytical models, cycle-accurate simulators, trace-driven approaches, and ML-augmented hybrid techniques. We survey 22 tools in depth (with 15+ additional tools discussed) drawn from 53 papers across architecture venues (MICRO, ISCA, HPCA, ASPLOS) and systems venues (MLSys, OSDI, NSDI) published between 2016–2026, spanning DNN accelerator modeling (Timeloop, MAESTRO, Sparseloop), GPU simulation (GPGPU-Sim, Accel-Sim, NeuSight), distributed training simulation (ASTRA-sim, Lumos, SimAI), and LLM inference serving (VIDUR, Frontier, AMALI). We organize the literature along three dimensions—methodology type (analytical, simulation, ML-augmented, hybrid), target platform (accelerators, GPUs, distributed systems, edge devices), and abstraction level (kernel, model, system)—while additionally characterizing tools by workload coverage, revealing a pervasive CNN-validation bias. Our analysis reveals that hybrid approaches combining analytical structure with learned components achieve the best accuracy-speed trade-offs, while pure analytical models offer superior interpretability for design space exploration. We conduct hands-on reproducibility evaluations of five representative tools, finding that reproducibility varies dramatically: among our evaluated tools, those with Docker-first deployment score 8.5+/10 on our rubric, while tools relying on serialized ML models (e.g., pickle-based predictors) risk becoming unusable due to dependency incompatibilities. We identify key open challenges including cross-workload generalization beyond CNNs, composition of kernel-level predictions to end-to-end accuracy, and support for emerging architectures. This survey provides practitioners guidance for selecting appropriate modeling tools and researchers a roadmap for advancing the field of ML workload performance prediction.

Keywords

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

MICRO 2026, Austin, TX, USA
2026.

1 Introduction

Machine learning workloads—spanning training and inference for CNNs, transformers, mixture-of-experts models, and graph neural networks—have become the dominant consumers of compute across datacenters and edge devices. The shift toward domain-specific architectures [25]—from Google's TPU [33, 34] to custom accelerators—has created a heterogeneous landscape where architects need fast, accurate performance predictions for design space exploration, parallelization selection, and hardware-software co-design. 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 tool ecosystem has emerged: analytical models (Timeloop [54]: 5–10% error; MAESTRO [42]: 5–15% error; both at microsecond speed), cycle-accurate simulators (GPGPU-Sim [4], Accel-Sim [37]: hours per workload), trace-driven simulators (ASTRA-sim [80], VIDUR [3]), and hybrid approaches (NeuSight [46]: 2.3% error). Yet no comprehensive survey organizes these methods for the practitioner who must select a tool for a specific task. Existing surveys focus on ML *techniques* for modeling [73] or specific hardware [54]; this survey fills that gap with a methodology-centric view.

We make the following contributions:

- A **methodology-centric taxonomy** organizing tools along three dimensions: methodology type (analytical, simulation, ML-augmented, hybrid), target platform, and abstraction level, with a coverage matrix identifying research gaps and a workload analysis exposing CNN-validation bias.
- A **systematic survey** of 22 tools (with 15+ additional tools discussed) from 53 papers across architecture venues (MICRO, ISCA, HPCA, ASPLOS) and systems venues (MLSys, OSDI, NSDI) published 2016–2026.
- A **comparative analysis** of accuracy–speed trade-offs with careful qualification of reported claims and cases where numbers are unverifiable.
- **Hands-on reproducibility evaluations** with a 10-point rubric, and identification of **open challenges** including the CNN-to-transformer generalization gap, kernel-to-end-to-end error composition, and emerging accelerator support.

The paper proceeds as follows: Section 2 describes our methodology; Section 3 provides background; Section 4 presents the taxonomy; Section 5 surveys tools; Section 6 compares accuracy; Section 7 presents evaluations; Section 8 discusses challenges; and Section 9 concludes. Figure 1 illustrates the evolution of performance modeling tools from early analytical frameworks to modern hybrid approaches.

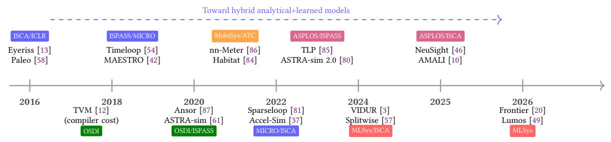

Figure 1: Evolution of performance modeling tools for ML workloads (2016–2026). Early analytical frameworks (Eyeriss, Paleo) gave way to systematic accelerator modeling (Timeloop, MAESTRO) and distributed training simulation (ASTRA-sim). Recent work targets LLM-specific modeling (VIDUR, Frontier) and hybrid analytical+learned approaches (NeuSight).

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, 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).

2.1 Related Surveys

Prior surveys address adjacent topics: Rakhshanfar and Zarandi [60] survey ML for processor DSE; Sze et al. [74] treat DNN hardware design (the foundation for Timeloop/MAESTRO); GPGPU-Sim [4] and gem5 [6] have extensive evaluation literature; SST [64] provides modular system-level simulation widely used for interconnect and HPC modeling; and MLPerf [51, 63] standardizes *measurement* rather than *prediction*. Early ML accelerator modeling (2014–2018) established foundational approaches: DianNao [11] introduced analytical dataflow modeling for dedicated accelerators, Eyeriss [13] systematized row-stationary dataflow analysis, and Paleo [58] pioneered layer-wise analytical estimation. This survey differs by spanning the full methodology spectrum across all major platforms with hands-on reproducibility evaluations. The closest prior work, Dudziak et al. [17], compares edge device predictors for NAS; we broaden to the full landscape.

3 Background

3.1 ML Workload Characteristics

ML workloads, defined as computation graphs in frameworks like PyTorch [56] and TensorFlow [1], have statically known operator shapes amenable to analytical modeling, though MoE and dynamic inference introduce input-dependent control flow. Performance depends on tensor-to-memory mapping (dataflow, tiling), KV cache management for LLM inference [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 [57], both modeled under batched serving [2, 83].

3.2 Modeling Methodologies

We classify approaches into five categories. **Analytical models** express performance as closed-form functions (e.g., the roofline model [79]: $P = \min(\pi, \beta \cdot I)$), offering microsecond evaluation but requiring per-architecture derivation. **Cycle-accurate simulators** (GPGPU-Sim [4], Accel-Sim [37]) achieve high fidelity at 1000–10000× slowdown [4, 37]; statistical sampling techniques [69, 82] and checkpoint-driven approaches [66] reduce this cost by identifying representative execution phases, while dedicated memory simulators [48, 50] provide cycle-accurate DRAM modeling. **Trace-driven simulators** (ASTRA-sim [80], VIDUR [3]) trade fidelity for orders-of-magnitude speedup. **ML-augmented approaches** learn from profiling data (nn-Meter [86]) but may not generalize beyond training distributions. **Hybrid approaches** combine analytical structure with learned components (NeuSight [46], Habitat [84]), aiming to balance accuracy, speed, and interpretability.

3.3 Problem Formulation

Performance modeling maps workload $\mathcal{W}$ and hardware $\mathcal{H}$ to a metric y: $\hat{y} = f(\mathcal{W}, \mathcal{H}; \theta)$, with workloads represented at operator, graph, IR, or trace level, and hardware characterized by specifications, counters, or learned embeddings. Prediction targets include latency, throughput, energy, and memory footprint; ground-truth measurements typically rely on hardware performance counters accessed via PAPI [7] or LIKWID [75]. Accuracy metrics—MAPE, RMSE, and rank correlation (Kendall's τ)—vary across the literature, and differences in benchmarks, hardware targets, and evaluation protocols limit direct comparison (Section 6).

4 Taxonomy

We organize the literature along three dimensions. The *primary axis* is methodology type—how a tool predicts performance—because methodology determines the fundamental trade-offs between accuracy, speed, interpretability, and data requirements. The *secondary axes* are target platform and abstraction level, which together determine the scope and applicability of each tool. We additionally characterize tools by workload coverage, exposing a pervasive CNN-validation bias in the literature.

Table 1 provides a unified view combining the coverage matrix (number of surveyed tools per methodology–platform cell) with trade-off profiles, with empty cells highlighting research gaps. 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.

Table 1 reveals three structural gaps: (1) trace-driven *execution replay* simulation (as distinct from instruction-trace-driven cycle-accurate simulation such as Accel-Sim) is used exclusively for distributed systems; (2) edge devices are served only by ML-augmented approaches, lacking hybrid alternatives; (3) no ML-augmented tool

Table 1: Methodology taxonomy: coverage matrix and trade-off profile. Platform columns show the number of surveyed tools per cell; 0 indicates an explicit research gap. Speed, data requirements, and interpretability determine practical applicability; the failure mode column identifies the primary condition under which each methodology breaks down.

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

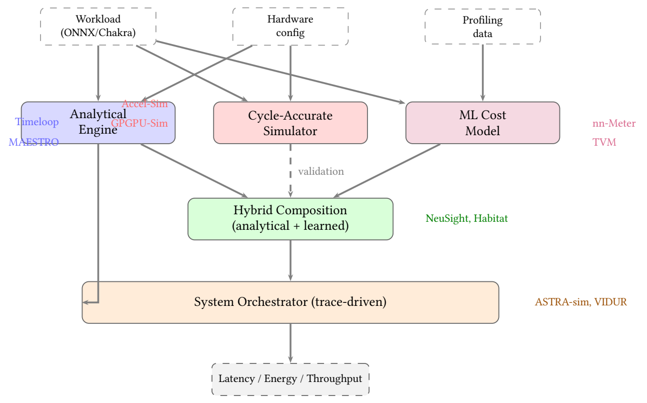

Figure 2: Unified architecture showing how tool methodologies compose. Analytical engines and ML cost models feed into hybrid approaches, while system-level orchestrators (trace-driven) assemble component predictions into end-to-end estimates. Cycle-accurate simulators primarily serve as validation oracles.

targets distributed systems directly. Methodologies cluster into two speed regimes: sub-millisecond (analytical, ML-augmented, hybrid) for DSE, and minutes-to-hours (simulation, trace-driven) for validation.

Figure 2 illustrates how tools from different methodology types compose: analytical engines provide fast base estimates, ML components learn residual corrections, and trace-driven simulators orchestrate system-level execution.

4.1 Primary Axis: Methodology Type

The choice of methodology determines fundamental trade-offs between accuracy, evaluation speed, data requirements, and interpretability, as summarized in Table 1; Section 5 provides detailed per-tool analysis.

Analytical models (Timeloop [54]: 5–10% vs. RTL; MAESTRO [42]; Sparseloop [81]; AMALI [10]) provide microsecond evaluation and full interpretability but require per-architecture derivation (AMALI's 23.6% MAPE illustrates GPU dynamic effects). **Cycle-accurate simulators** (GPGPU-Sim [4], Accel-Sim [37]: 0.90–0.97 IPC; PyTorchSim [38]) are impractical for DSE at 1000–10000× slowdown [4, 37]. **Trace-driven simulators** (ASTRA-sim [80]: 5–15%; VIDUR [3]: <5%; SimAI [77]; Frontier [20]) replay execution traces for system-level modeling. **ML-augmented models** (nn-Meter [86]; LitePred [18]; HELP [45]; TVM [12]/Ansor [87]) learn from profiling data but risk *silent distribution shift*. **Hybrid** approaches (NeuSight [46]: 2.3% MAPE; Habitat [84]; ArchGym [41]) combine analytical priors with learned corrections [17].

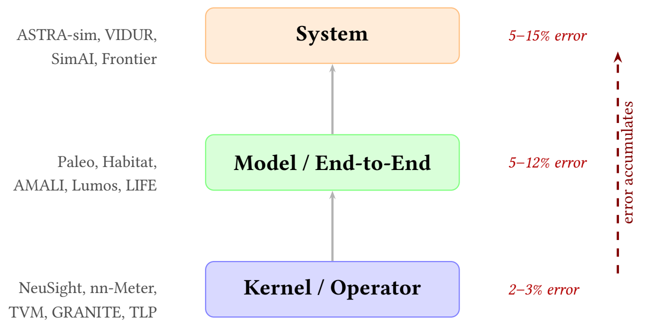
ASTRA-sim, VIDUR, SimAI, Frontier
System
5–15% error
Paleo, Habitat, AMALI, Lumos, LIFE
Model / End-to-End
5–12% error
NeuSight, nn-Meter, TVM, GRANITE, TLP
Kernel / Operator
2–3% error
error accumulates

Figure 3: Abstraction level hierarchy and the composition problem. Tools operate at one of three levels; composing predictions across levels accumulates error. Error ranges are representative values from surveyed papers.

4.2 Secondary Axes: Platform and Abstraction Level

Platform constrains methodology: **accelerators** use analytical models; **GPUs** span all types; **distributed systems** require trace-driven simulation; **edge devices** use ML-augmented approaches; **CPUs** [53, 73] are least studied. Abstraction level determines composition errors (Figure 3): kernel-level tools achieve 2–3% error, model-level 5–12%, and system-level 5–15%, with errors propagating through the chain.

4.3 Workload Coverage

Table 2 characterizes the workload types on which each tool has been validated, exposing a pervasive CNN-validation bias.

Figure 4 quantifies this CNN-validation bias: of the 14 surveyed tools, 10 (71%) include CNN validation, while only 1 tool validates on MoE workloads and none validates on diffusion models. The table reveals that **no surveyed tool has been validated on diffusion models or dynamic inference workloads** [39], only Frontier [20] has validated MoE support, and no single tool offers

Table 2: Workload validation coverage. ✓ = validated in the original paper; ○ = partial or indirect validation; — = no validation. Nearly all tools report accuracy on CNN workloads; transformer and MoE coverage is sparse. Empty columns (diffusion, dynamic inference) represent workload types with *no* validated performance modeling tools.

Tool	CNN	Trans-former	LLM Train	MoE	Diff.
Timeloop	✓	○	—	—	—
MAESTRO	✓	—	—	—	—
NeuSight	✓	✓	—	—	—
Habitat	✓	—	—	—	—
AMALI	—	✓	—	—	—
ASTRA-sim	✓	○	✓	—	—
VIDUR	—	✓	—	—	—
SimAI	—	—	✓	—	—
Lumos	—	—	✓	—	—
Frontier	—	✓	—	✓	—
nn-Meter	✓	—	—	—	—
LitePred	✓	—	—	—	—
HELP	✓	—	—	—	—
TVM/Ansor	✓	○	—	—	—

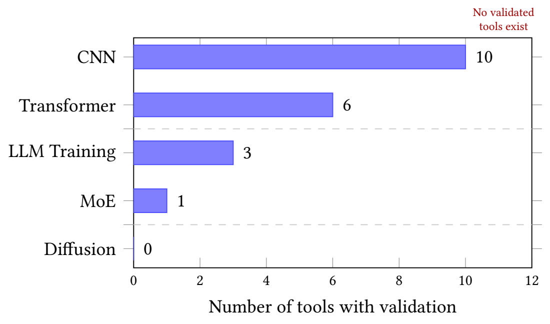

Figure 4: Workload validation coverage across surveyed tools. CNN validation dominates (71% of tools), while MoE and diffusion models have minimal or no validated prediction tools, highlighting a critical generalization gap.

validated transformer prediction across the full kernel-to-system stack. Practitioners working with non-CNN workloads must accept unvalidated predictions, collect their own validation data, or fall back to measurement.

5 Survey of Approaches

This section surveys performance modeling tools for ML workloads, organized by target platform, examining modeling challenges, available tools, and their strengths and limitations. Table 3 provides a comprehensive comparison.

5.1 DNN Accelerator Modeling

DNN accelerators' computational regularity—fixed dataflow patterns and deterministic memory access—makes this domain uniquely amenable to analytical modeling [74], building on early accelerator characterization pioneered by DianNao [11]. Timeloop [54] exploits this regularity by enumerating loop-nest orderings that determine data reuse: because each loop permutation maps to a unique set of spatial and temporal data movements, Timeloop can analytically compute the exact number of memory accesses at each hierarchy level, achieving 5–10% error with 2000× speedup. MAESTRO [42] raises the abstraction level with data-centric directives that describe *which* data dimensions are spatially partitioned rather than *how* loops are ordered; this enables fast operator-level analysis (μs per mapping) because MAESTRO avoids enumerating individual loop nests and instead reasons about data reuse algebraically. The trade-off: MAESTRO's abstraction sacrifices Timeloop's ability to model per-PE utilization, explaining why Timeloop achieves tighter error bounds on architectures with irregular PE arrays. SCALE-Sim [67] complements both by providing cycle-accurate systolic array simulation for validation. Sparseloop [81] extends the loop-nest framework to sparse tensors, modeling compression formats (CSR, bitmap) and their impact on irregular memory accesses. PyTorchSim [38] integrates PyTorch 2 with NPU simulation but lacks real-hardware validation; ArchGym [41] connects ML surrogates to simulators (0.61% RMSE vs. simulator, not hardware). Accelerator modeling is the most mature subdomain, with Timeloop as the de facto DSE standard. The key gap is silicon validation; emerging PIM tools [26, 31, 44, 55] also lack hardware validation.

5.2 GPU Performance Modeling

GPUs dominate ML training and inference, requiring models for SIMT execution, warp scheduling, memory coalescing, and occupancy effects.

Cycle-accurate simulation. GPGPU-Sim [4] and Accel-Sim [37] achieve 0.90–0.97 IPC correlation but at 1000–10000× slowdown; reverse-engineering [30] improved Accel-Sim to 13.98% MAPE. These simulators integrate with memory subsystem models—from DRAMSim2 [65] and Ramulator [40] to their modern successors DRAMSim3 [48] and Ramulator 2.0 [50]—for accurate DRAM timing, critical for memory-bound LLM inference.

Analytical and hybrid models. AMALI [10] models GPU performance through memory hierarchy analysis—computing data movement volumes across L1, L2, and HBM levels—but its 23.6% MAPE reveals a fundamental limitation: analytical memory models cannot capture dynamic effects (warp scheduling contention, bank conflicts, memory coalescing efficiency) that dominate real GPU execution, especially for attention kernels with irregular access patterns. The roofline model [79] provides upper bounds, with recent extensions adapting it to LLM-specific workloads [32]. NeuSight [46] achieves 2.3% on GPT-3 by decomposing kernels into tiles that mirror CUDA's actual thread block tiling: because GPU hardware schedules thread blocks as atomic units with shared memory, tile-level predictions naturally capture occupancy effects and memory locality that per-operator models miss. This structural alignment with hardware execution—predicting per-tile latency and aggregating—is why NeuSight achieves 10× lower error than AMALI despite both being sub-second tools. Habitat [84] achieves 11.8% cross-GPU transfer via wave scaling.

Table 3: Summary of surveyed performance modeling tools for ML workloads, organized by target platform. Methodology: A=Analytical, S=Simulation, T=Trace-driven, M=ML-augmented, H=Hybrid. *Accuracy measures surrogate-vs-simulator fidelity, not real hardware error. †Reported accuracy unverifiable due to reproducibility issues. ‡No accuracy baseline against real hardware reported.

Tool	Platform	Method	Target	Accuracy	Speed	Key Capability
DNN Accelerator Modeling						
Timeloop [54]	NPU	A	Latency/Energy	5–10%	μs	Loop-nest DSE
MAESTRO [42]	NPU	A	Latency/Energy	5–15%	μs	Data-centric directives
Sparseloop [81]	NPU	A	Sparse tensors	5–10%	μs	Compression modeling
PyTorchSim [38]	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 [37]	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 [46]	GPU	H	Kernel/E2E latency	2.3%	ms	Tile-based prediction
Habitat [84]	GPU	H	Training time	11.8%	Per-kernel	Wave scaling
Distributed Training and LLM Serving						
ASTRA-sim [80]	Distributed	T	Training time	5–15%	Minutes	Collective modeling
SimAI [77]	Distributed	T	Training time	1.9%	Minutes	Full-stack simulation
Lumos [49]	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 [47]	Multi-GPU	T	DNN training	N/A‡	Minutes	Lightweight multi-GPU
Edge Device Modeling						
nn-Meter [86]	Edge	M	Latency	<1%†	ms	Kernel detection
LitePred [18]	Edge	M	Latency	0.7%	ms	85-platform transfer
HELP [45]	Multi	M	Latency	1.9%	ms	10-sample adaptation
Compiler Cost Models						
TVM [12]	GPU	M	Schedule perf.	~15%	ms	Autotuning guidance
Ansor [87]	GPU	M	Schedule perf.	~15%	ms	Program sampling
TLP [85]	GPU	M	Tensor program	<10%	ms	Transformer cost model

LLM-specific and compiler models. VIDUR [3] simulates LLM serving with scheduling strategies at <5% error; LIFE [19] and HERMES [5] target inference; Omniwise [23] and SwizzlePerf [76] are emerging predictors. TVM [12]/Ansor [87] (~15% MAPE), TLP [85] (<10%), and SynPerf [78] target compiler autotuning [88]. GPU modeling spans 2%–24% error; NeuSight offers the best accuracy–speed trade-off for LLMs, while AMALI fills pre-silicon gaps.

5.3 Distributed Training and LLM Serving

Distributed systems require modeling communication, synchronization, and parallelism strategies [29, 59, 70]. ASTRA-sim [80] achieves 5–15% error via Chakra traces [71]; SimAI [77] reaches 1.9% at Alibaba scale; Echo [8] scales simulation to 10K+ devices; Lumos [49] 3.3% on H100s; PRISM [21] provides prediction intervals at 10K+ GPUs. Paleo [58] pioneered analytical estimation; MAD Max [28] and Sailor [72] extend it; Llama 3 [15] provides validation ground truth at 16K GPUs. For inference serving, VIDUR [3] models scheduling with vLLM [43]; DistServe [89] disaggregates prefill and decode for goodput optimization; Frontier [20] targets MoE; POD-Attention [24] and AQUA [68] address prefill-decode overlap and memory offloading respectively; ThrottLL'eM [35] models power effects; speculative decoding [9] creates a moving target for all simulators.

5.4 Edge Device Modeling

Hardware diversity makes per-device analytical modeling impractical. nn-Meter [86] claims <1% MAPE but is unverifiable (3/10 reproducibility, Section 7); LitePred [18] achieves 0.7% across 85 platforms; HELP [45] reaches 1.9% with 10-sample meta-learning. ESM [52] 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.

5.5 Cross-Cutting Themes

Three architectural insights emerge from our analysis.

First, structural decomposition that mirrors hardware execution consistently outperforms black-box approaches. Timeloop's loop nests reflect systolic array dataflow, NeuSight's tiles mirror CUDA thread block scheduling, and VIDUR's prefill/decode split captures the GPU's distinct compute- vs. memory-bound regimes. In each case, the modeling abstraction succeeds because it aligns with the hardware boundary that dominates performance: data reuse at the PE array level, occupancy at the SM level, and phase-level batching at the system level. By contrast, AMALI's single-pass memory hierarchy model misses dynamic scheduling effects that cross these boundaries, explaining its higher error. Notably, tools with *verifiable* accuracy (e.g., Timeloop's reference outputs, VIDUR's Docker-based reproduction) appear more widely adopted than tools reporting high but unverifiable accuracy (e.g., nn-Meter).

Second, the most critical architectural features for accurate ML modeling differ by platform. For accelerators, data reuse (determined by dataflow and tiling) dominates because accelerator designs deliberately eliminate dynamic effects. For GPUs, thread block occupancy and memory coalescing are most critical because the SIMT execution model introduces contention not present in systolic arrays. For distributed systems, collective communication topology and pipeline bubble overhead dominate because compute per device is well-characterized but inter-device interaction is not. This explains why no single methodology works across all platforms.

Third, a persistent ***accuracy–generality–speed trade-off*** explains methodological diversity: cycle-accurate simulators (Accel-Sim, GPGPU-Sim) maximize accuracy by modeling microarchitectural state but sacrifice speed; analytical models (Timeloop, MAESTRO) maximize speed by exploiting structural regularity but sacrifice accuracy on dynamic workloads; ML-augmented approaches (nn-Meter, LitePred) achieve both speed and accuracy on trained distributions but sacrifice generality to unseen architectures. Subdomain maturity correlates with economic incentive: accelerator DSE is most mature (irreversible chip errors), distributed training is fastest-growing (million-dollar runs), and edge modeling has weakest reproducibility.

6 Comparison and Analysis

We analyze trade-offs across methodology types along accuracy and speed dimensions (see Table 3 for per-tool details); generalization and interpretability challenges are deferred to Section 8. Figure 5 visualizes the accuracy–speed trade-off space, revealing three distinct clusters of tools.

6.1 Accuracy by Problem Difficulty

We organize accuracy results by inherent problem difficulty rather than comparing across incompatible benchmarks (Figure 6). Accelerator dataflow modeling is most tractable (Timeloop: 5–10%) because systolic arrays exhibit deterministic data movement—the absence of dynamic scheduling, speculative execution, and shared caches eliminates the primary sources of modeling error. Single-GPU kernel prediction achieves 2–12% via hybrid methods (NeuSight, Habitat), where the accuracy gap relative to accelerators arises from GPU-specific dynamic effects: warp scheduling decisions that depend on runtime register pressure, memory coalescing efficiency that varies with access patterns, and shared memory bank conflicts that are input-dependent. Distributed systems reach 2–15% (SimAI 1.9%, ASTRA-sim 5–15%), where the dominant error source shifts from compute modeling to communication topology and collective algorithm selection. Cross-platform edge prediction achieves 0.7–2% but requires per-device profiling; GPU analytical modeling remains hardest (AMALI: 23.6%). Setup costs vary dramatically: analytical models require only architecture specifications,

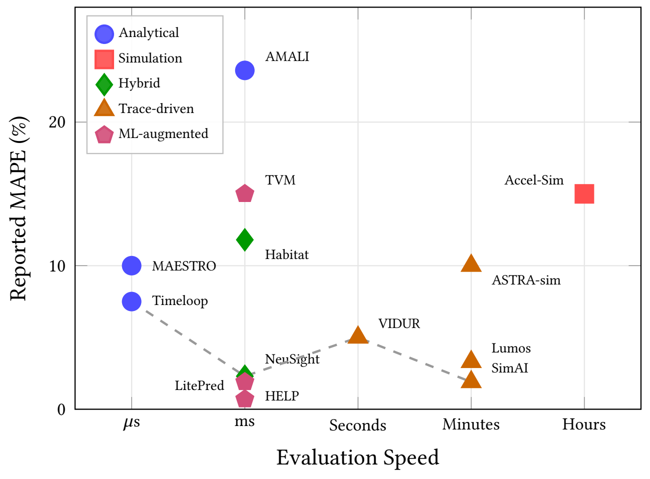

Figure 5: Accuracy–speed trade-off across surveyed tools. Each point represents a tool's reported MAPE vs. evaluation speed (log-scale categories). The dashed line traces the approximate Pareto frontier. Hybrid (NeuSight) and trace-driven (SimAI) approaches dominate the frontier.

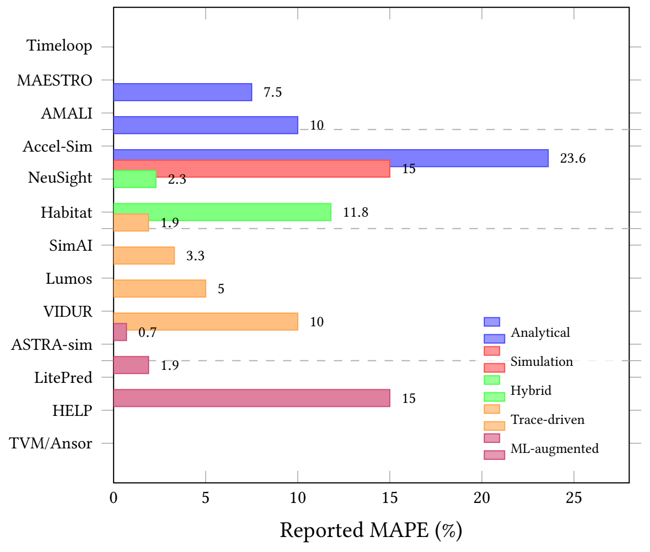

Figure 6: Reported accuracy (MAPE) of surveyed tools, grouped by methodology type. Range midpoints used where ranges are reported. Cross-tool comparison is approximate due to differing benchmarks, workloads, and hardware targets.

ML-augmented approaches need 10–10K profiling samples per device, and cycle-accurate simulators require hardware-specific binaries or traces.

6.2 Practitioner Tool Selection

Tool selection depends on the target platform, acceptable error margin, and available setup time; Figure 7 provides a decision flowchart.

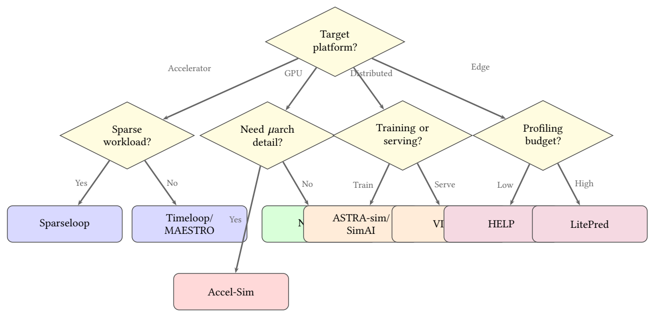

Figure 7: Tool selection decision flowchart. Practitioners choose based on target platform, then refine by workload characteristics and resource constraints. Colors indicate methodology type: blue=analytical, green=hybrid, orange=trace-driven, purple=ML-augmented.

Table 4: Reproducibility evaluation scores (10-point rubric). Tools are ranked by total score. †Timeloop CLI works but Python bindings fail.

Tool	Setup	Reprod.	Usability	Total
VIDUR	2.5	3.5	3	9/10
Timeloop†	3	4	2	9/10
ASTRA-sim	2.5	3	3	8.5/10
NeuSight	2	3	2.5	7.5/10
nn-Meter	2	0	1	3/10

For *accelerator DSE*, use Timeloop or MAESTRO for microsecond-speed exhaustive search with interpretable bottleneck feedback; Sparseloop extends this to sparse workloads. For *GPU evaluation*, NeuSight offers the best accuracy–speed balance for LLMs; use Accel-Sim when microarchitectural detail is needed, accepting the 1000× slowdown. For *distributed systems*, use VIDUR for LLM serving configuration and ASTRA-sim or SimAI for training parallelism at scale; MAD Max provides fast analytical estimates when trace collection is impractical. For *edge devices*, LitePred offers the broadest platform coverage, while HELP excels with minimal profiling data. Practitioners should consider tools with Docker-first deployment (VIDUR, Timeloop, ASTRA-sim) over tools with unpinned dependencies, as our evaluation of five tools suggests containerization correlates with higher reproducibility scores.

7 Experimental Evaluation

We conducted hands-on evaluations of five tools spanning methodology types: Timeloop (analytical), ASTRA-sim (trace-driven, distributed), VIDUR (trace-driven, LLM serving), nn-Meter (ML-augmented, edge), and NeuSight (hybrid, GPU).

Environment and rubric. All evaluations ran on Apple M2 Ultra (aarch64, 192 GB RAM) using Docker containers where provided—no GPU hardware was available, so we cannot validate absolute accuracy claims. We score each tool on a 10-point rubric: *Setup* (3 pts), *Reproducibility* (4 pts), *Usability* (3 pts). Table 4 summarizes results.

Table 5: VIDUR simulation results for Llama-2-7B inference serving on a simulated A100 GPU. All metrics from our own experiments.

Metric	vLLM	Sarathi
Requests	200	50
Avg E2E latency (s)	0.177	0.158
P99 E2E latency (s)	0.320	0.270
Avg TTFT (s)	0.027	0.025
Avg TPOT (s)	0.0093	0.0090
Requests preempted	53	0

Table 6: ASTRA-sim quantitative results from our experiments on the HGX-H100 configuration. Top: collective microbenchmarks (8 NPUs, 1 MB). Bottom: ResNet-50 data-parallel training 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%

7.1 Per-Tool Results

VIDUR (9/10). We simulated Llama-2-7B on a simulated A100 (Table 5). Sarathi achieves 12.2% lower latency than vLLM via chunked prefill [2]; TPOT differs by only 3.5%; vLLM preempted 26.5% of requests vs. zero for Sarathi—matching KV-cache management differences [43].

Timeloop (9/10). Docker CLI produces deterministic, bit-identical outputs for Eyeriss-like configurations; reference outputs enable hardware-free verification. Python bindings fail (`ImportError: libbarvinok.so.23`).

ASTRA-sim (8.5/10). We ran collective microbenchmarks and ResNet-50 training at 2–8 GPUs (Table 6). Reduce-Scatter takes half the time of All-Reduce (consistent with half the data); communication overhead scales 5.76× for 4× more GPUs, matching ring All-Reduce scaling.

NeuSight (7.5/10). Tile-based decomposition mirrors CUDA tiling for dense operations; irregular workloads had limited examples.

nn-Meter (3/10). After four attempts (>4h), no predictions ran: pickle-serialized predictors (scikit-learn 0.23.1) are incompatible with current versions. The claimed <1% MAPE is **unverifiable**.

Figure 8 visualizes the reproducibility scores, showing that Docker-first tools scored consistently higher in our evaluation.

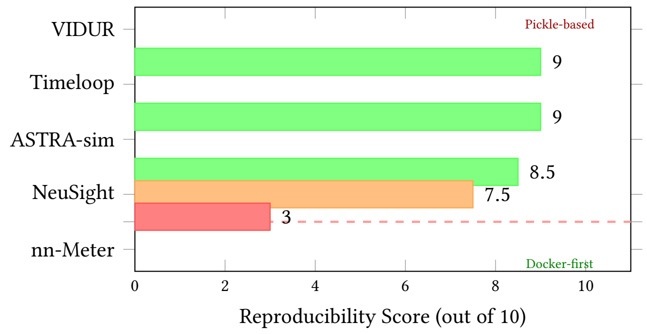

Figure 8: Reproducibility scores for evaluated tools. Docker-first tools (VIDUR, Timeloop, ASTRA-sim) consistently score 8.5+/10, while tools relying on serialized ML models (nn-Meter) become unusable. The dashed line separates Docker-based from non-Docker deployments.

7.2 Lessons and Threats to Validity

Five lessons emerge: (1) **Docker-first deployment** correlates with high reproducibility in our sample (Docker tools: 8.5+/10; nn-Meter without Docker: 3/10), though our sample of five tools limits causal conclusions. (2) **ML model serialization is fragile**—nn-Meter's pickle-based predictors became unusable within two years. (3) **Reference outputs enable trust without hardware**—Timeloop and ASTRA-sim include verifiable baselines. (4) **Scale-limited evaluation understates system tools**—our 2–8 GPU tests show only 0.30% communication overhead, far below production scales [15]. (5) **Reproducible accuracy claims should be weighted higher** than unreproducible ones.

Threats. Our venue-focused search may under-represent industry and non-English publications; we exclude proprietary tools (Nsight Compute, internal TPU models); and accuracy metrics vary across papers (MAPE, RMSE, Kendall's τ), limiting direct comparison.

8 Open Challenges and Future Directions

Generalization gaps. *Workload*: CNN→transformer transfer is largely unvalidated (NeuSight excepted); MoE, diffusion [39], and dynamic inference lack validated tools; scaling laws [14, 22, 27, 36] predict loss but not latency. Figure 9 shows the shift toward LLM workloads since 2023. *Hardware*: cross-family transfer (GPU→TPU→PIM) remains unsolved despite meta-learning (HELP) and feature-based transfer (LitePred). *Temporal*: software stack evolution silently invalidating models is addressed by no tool.

The composition problem. Composing kernel-level predictions into end-to-end estimates is unsolved (Figure 10): NeuSight's 2.3% kernel MAPE yields ~10× higher variance at model level ($\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$), and correlated errors can compound linearly. VIDUR sidesteps this by profiling entire prefill/decode phases.

Emerging hardware and reproducibility. PIM [26, 31, 44, 55], chiplets, and disaggregated designs blur memory hierarchy assumptions; FlashAttention [16] changes the landscape faster than models retrain. No MLPerf [51, 63] equivalent exists for performance *prediction.*

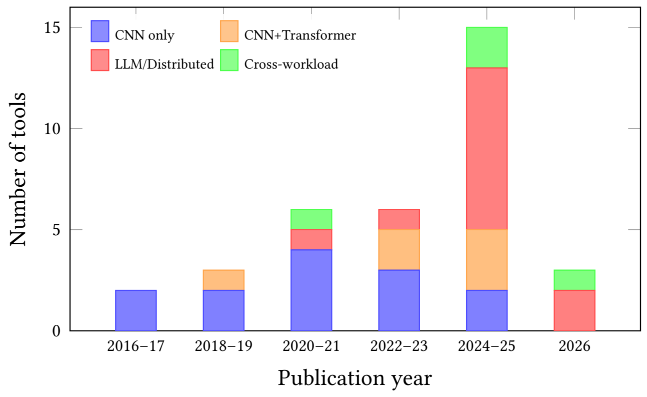

Figure 9: Workload coverage of surveyed tools by publication period. The shift toward transformer and LLM workloads accelerates from 2023, but MoE and diffusion models remain largely uncharacterized.

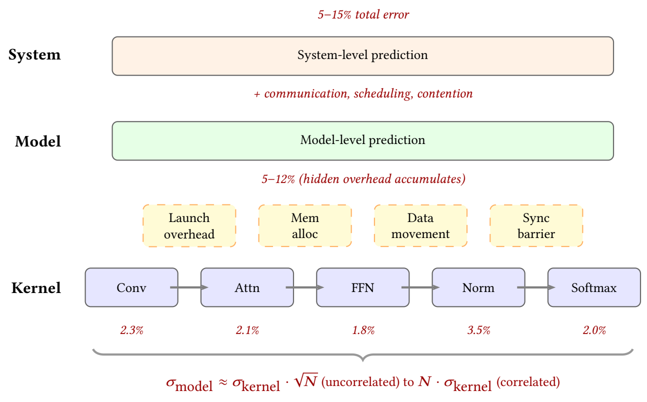
5–15% total error
System
System-level prediction
+ communication, scheduling, contention
Model
Model-level prediction
5–12% (hidden overhead accumulates)
Launch overhead
Mem alloc
Data movement
Sync barrier
Kernel
Conv
Attn
FFN
Norm
Softmax
2.3%
2.1%
1.8%
3.5%
2.0%
$\sigma_{\text{model}} \approx \sigma_{\text{kernel}} \cdot \sqrt{N}$ (uncorrelated) to $N \cdot \sigma_{\text{kernel}}$ (correlated)

Figure 10: Error composition across abstraction levels. Kernel-level predictions (2–3% each) accumulate through hidden overheads (kernel launch, memory allocation, data movement, synchronization) that are not captured by kernel-level tools, yielding 5–12% model-level error. System-level errors add communication and scheduling overhead.

Future directions: (1) validated non-CNN tools; (2) bounded composition error; (3) unified energy-latency-memory prediction [62]; (4) temporal robustness benchmarks; (5) Docker-first deployment with portable formats (ONNX, Chakra [71]).

9 Conclusion

This survey analyzed 22 tools in depth for predicting ML workload performance, organized by methodology type, target platform, and abstraction level. Key findings: (1) *No single methodology dominates*—analytical models offer microsecond interpretable evaluation, trace-driven simulators provide 2–15% system-level error, and hybrid approaches achieve the best accuracy–speed balance (NeuSight: 2.3% MAPE), with the right choice depending on the practitioner's priorities. (2) *LLM workloads demand specialized modeling*—prefill/decode distinctions, KV cache management, and dynamic batching require purpose-built tools (VIDUR, Frontier) rather than CNN-era extensions. (3) *Reproducibility is a practical bottleneck*—Docker-first tools score 8.5+/10 while tools relying on serialized ML models have become unusable. (4) *Accuracy claims require scrutiny* due to varying benchmarks and metrics.

The most pressing gaps are CNN-to-transformer generalization, kernel-to-end-to-end composition, emerging hardware support (PIM, chiplets), and reproducibility failures. As ML workloads grow in scale and diversity, this survey provides practitioners guidance for tool selection and researchers a roadmap for advancing the field.

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] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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

[38] 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.

[39] 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.

[40] Yoongu Kim, Weikun Yang, and Onur Mutlu. 2016. Ramulator: A Fast and Extensible DRAM Simulator. *IEEE Computer Architecture Letters* 15, 1 (2016), 45–49. https://doi.org/10.1109/LCA.2015.2414456 Fast extensible DRAM simulator supporting DDRx, LPDDRx, GDDRx, WIOx, HBMx standards.

[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] Hyojung Lee, Daehyeon Baek, Jimyoung 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..

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] Haocong Luo, Yahya Can Tugrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkcı, 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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

[55] 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..

[56] 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.

[57] 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.

[58] 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

[59] 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.

[60] 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

[61] 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

[62] 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.

[63] 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.

[64] 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.

[65] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. 2011. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letters* 10, 1 (2011), 16–19. https://doi.org/10.1109/L-CA.2011.4 Widely-used cycle-accurate DDR2/DDR3 memory simulator validated against manufacturer Verilog models.

[66] Alen Sabu, Harish Patil, Ameer Haj-Ali, and Trevor E. Carlson. 2022. LoopPoint: Checkpoint-driven Sampled Simulation for Multi-threaded Applications. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 606–618. https://doi.org/10.1109/HPCA53966.2022.00052 Extends sampling to multi-threaded applications with 2.3% error and up to 800x speedup.

[67] 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.

[68] 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.

[69] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. 2002. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 45–57. https://doi.org/10.1145/605397.605403 Introduces SimPoint: automatic selection of representative simulation points using k-means clustering.

[70] 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.

[71] 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).

[72] 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..

[73] 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

[74] 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.

[75] 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.

[76] 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.

[77] 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..

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

[79] 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

[80] 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

[81] 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

[82] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. 2003. SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling. In *Proceedings of the 30th Annual International Symposium on Computer Architecture (ISCA)*. 84–97. https://doi.org/10.1109/ISCA.2003.1206991 Statistical sampling achieving 0.64% CPI error with 35x speedup over detailed simulation.

[83] 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.

[84] 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.

[85] 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

[86] 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.

[87] 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.

[88] 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.

[89] 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.