

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 approximately 25 tools 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: Docker-first tools score 8.5+/10 on our rubric while tools relying on serialized ML models risk becoming unusable. 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

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 [22], from Google’s TPU [30, 31] to custom training accelerators, has created a heterogeneous hardware landscape where architects and system designers need fast, accurate performance predictions to navigate vast design spaces, select parallelization strategies, provision serving infrastructure, and optimize hardware-software co-design. Yet ML workloads pose unique modeling challenges: they exhibit diverse computational patterns (dense matrix operations in attention layers, sparse accesses in GNNs, communication-bound collective operations in distributed training) across this increasingly heterogeneous landscape of GPUs, TPUs, custom accelerators, and multi-device clusters.

A rich ecosystem of tools has emerged, spanning analytical models (Timeloop [48], MAESTRO [38]: 5–10% error at microsecond speed), cycle-accurate simulators (GPGPU-Sim [4], Accel-Sim [34]: detailed but hours per workload), trace-driven simulators (ASTRA-sim [68], VIDUR [3]: system-scale training and serving), and ML-augmented hybrid approaches (NeuSight [42]: 2.3% error on GPU kernels). Each methodology occupies a distinct point in the accuracy-speed-generality trade-off space.

Despite this rich tool landscape, no comprehensive survey organizes these methods from the perspective of the ML workload practitioner—the architect or engineer who needs to select a modeling tool for a specific design or deployment task. Existing surveys focus on ML *techniques* for performance modeling [62] or on specific hardware targets [48], leaving practitioners without guidance on which tools suit their needs across the full modeling spectrum. This survey fills that gap by providing a methodology-centric view of the tools and methods available for predicting ML workload performance.

We make the following contributions:

- A **methodology-centric taxonomy** organizing tools along three dimensions: methodology type (analytical, simulation, ML-augmented, hybrid), target platform (DNN accelerators, GPUs, distributed systems, edge devices), and abstraction level (kernel, model, system), with a quantitative coverage matrix identifying research gaps and a workload coverage analysis exposing the CNN-validation bias in the literature.
- A **systematic survey** of over 30 modeling tools drawn from 53 papers across architecture venues (MICRO, ISCA, HPCA, ASPLOS) and systems venues (MLSys, OSDI, NSDI) published between 2016–2026, using documented selection criteria.
- A **comparative analysis** examining trade-offs between accuracy, speed, generalization, and interpretability, with careful qualification of paper-reported accuracy claims and

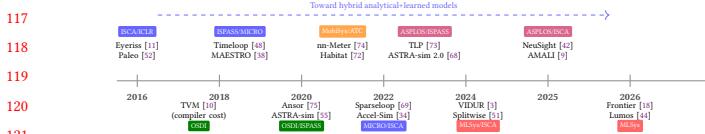


Figure 1: Evolution of performance modeling tools for ML workloads (2016–2026). Early analytical frameworks (EyeRISS, PALEO) gave way to systematic accelerator modeling (Timeloop, MAESTRO) and distributed training simulation (ASTRA-sim). Recent work targets LLM-specific modeling (VIDUR, FRONTIER) and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

identification of cases where reported numbers are unverifiable.

- **Hands-on reproducibility evaluations** of representative tools 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 by platform; Section 6 compares accuracy and provides tool selection guidance; Section 7 presents reproducibility evaluations; Section 8 discusses open challenges; and Section 9 concludes.

Figure 1 illustrates the evolution of performance modeling tools for ML workloads, from early analytical frameworks through simulators 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, 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 [54] survey ML for processor DSE; Sze et al. [63] treat DNN hardware design (the foundation for Timeloop/MAESTRO); GPGPU-Sim [4] and gem5 [6] have extensive evaluation literature; and MLPerf [45, 57] standardizes *measurement* rather than *prediction*. 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. [15], 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 [50] and TensorFlow [1], present distinct modeling challenges. Their operators (convolutions, matrix multiplications, attention) have statically known shapes amenable to analytical modeling, though mixture-of-experts and dynamic inference introduce input-dependent control flow. Performance is highly sensitive to how tensors map onto specialized memory hierarchies (dataflow, tiling, loop ordering), and for LLM inference, KV cache management dominates memory behavior [39]. At scale, training distributes across thousands of GPUs via data, tensor, pipeline, and expert parallelism [13], requiring system-level modeling of compute–memory–network interactions. LLM inference further splits into compute-bound prefill and memory-bound decode phases [51], both of which must be modeled under batched serving [2, 71].

3.2 Modeling Methodologies

We classify approaches into four categories forming our taxonomy’s primary axis. **Analytical models** express performance as closed-form functions—e.g., the roofline model [67] bounds throughput by $P = \min(\pi, \beta \cdot I)$, while Timeloop [48] computes data movement costs for DNN accelerator mappings. They offer microsecond evaluation but require per-architecture derivation. **Cycle-accurate simulators** (gem5 [6], GPGPU-Sim [4], Accel-Sim [34]) achieve high fidelity but at 1000–10000× slowdown; sampling techniques [58, 70] help but were not designed for ML workloads. **Trace-driven simulators** like ASTRA-sim [68] (distributed training via Chakra traces [60]) and VIDUR [3] (LLM serving) trade some fidelity for orders-of-magnitude speedup. **ML-augmented approaches** learn performance functions from profiling data, ranging from random forests (nn-Meter [74]) and XGBoost (TVM [10]) to deep learning (NeuSIGHT [42]) and meta-learning (HELP [41]); they capture nonlinear relationships but may not generalize beyond their training distribution.

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. 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 observations. First, trace-driven simulation is exclusively used for distributed systems—no surveyed tool applies trace-driven methods to single-device GPU or accelerator modeling, despite the potential for trace-driven approaches to avoid the slowdown of cycle-accurate simulation while retaining more fidelity than analytical models. Second, edge/mobile devices are served exclusively by ML-augmented approaches; the absence of analytical or hybrid models for edge devices reflects the hardware diversity problem but also represents a research gap, since hybrid approaches could combine the interpretability of analytical models with the adaptability of learned components. Third, no ML-augmented or hybrid tool specifically targets distributed system modeling—tools like VIDUR use ML internally for kernel prediction but are architecturally trace-driven simulators. The trade-off columns further show that methodologies cluster into two speed regimes: sub-millisecond (analytical, ML-augmented, hybrid) suitable for design space exploration, and minutes-to-hours (simulation, trace-driven) suitable for detailed validation.

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 express performance as closed-form functions of workload and hardware parameters. Timeloop [48] achieves 5–10% error versus RTL at 2000× speedup for DNN accelerators; MAESTRO [38] and Sparseloop [69] extend to data-centric directives and sparse tensors; Paleo [52] and AMALI [9] target distributed training and GPU LLM inference, respectively. These models provide microsecond evaluation and full interpretability but require per-architecture derivation and may miss dynamic effects (AMALI’s 23.6% MAPE illustrates this ceiling for GPUs).

Cycle-accurate simulators model hardware at register-transfer level. GPGPU-Sim [4] and Accel-Sim [34] achieve 0.90–0.97 IPC correlation; PyTorchSim [35] integrates PyTorch 2 with NPU simulation. Speed (1000–10000× slowdown) makes these impractical for ML design space exploration.

Trace-driven simulators replay execution traces for system-level modeling. ASTRA-sim [68] achieves 5–15% error for distributed training via Chakra traces [60]; VIDUR [3] provides <5% error for LLM serving; SimAI [65], Lumos [44], and Frontier [18] target large-scale training and MoE inference. Some tools use ML internally (e.g., VIDUR’s random forests for kernel prediction), blurring the boundary with hybrid approaches.

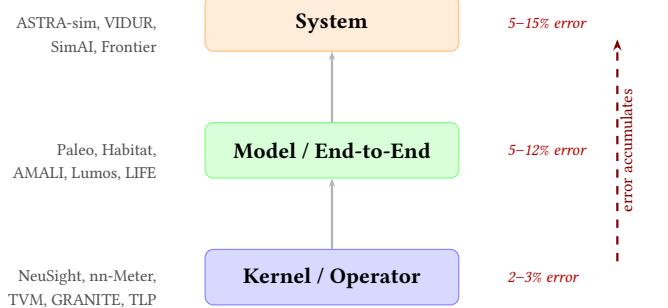


Figure 2: Abstraction level hierarchy and the composition problem. Tools operate at one of three levels; composing predictions across levels accumulates error. Error ranges are representative values from surveyed papers.

ML-augmented models learn performance functions from profiling data: nn-Meter [74] (random forests for edge devices), LitePred [16] (85-platform transfer), HELP [41] (10-sample meta-learning), and TVM [10]/Ansor [75] (compiler autotuning). Their critical failure mode is *silent distribution shift*—CNN-trained models may produce confident but wrong predictions for transformers.

Hybrid analytical+ML models combine physics-based priors with learned residual corrections. NeuSight [42] achieves 2.3% MAPE on GPT-3 inference via tile-based prediction; Habitat [72] decomposes compute/memory components; ArchGym [37] connects ML surrogates to analytical simulators. Transfer learning provides 22.5% average improvement [15].

4.2 Secondary Axes: Platform and Abstraction Level

The target platform constrains applicable methodologies: **DNN accelerators** [30, 31] are best served by analytical models (Timeloop, MAESTRO, Sparseloop) due to regular memory hierarchies; **GPUs** span the full methodology spectrum reflecting SIMD complexity; **distributed systems** require trace-driven simulation (ASTRA-sim, VIDUR, SimAI) for collective communication and pipeline parallelism; **edge/mobile devices** are dominated by ML-augmented approaches (nn-Meter, LitePred, HELP) due to hardware diversity; and **CPUs** are less studied for ML workloads (Concorde [47], GRANITE [62]).

Abstraction level determines where composition errors arise. **Kernel-level** tools (NeuSight, nn-Meter, TVM) achieve 2–3% error but composing predictions into end-to-end latency introduces errors from memory allocation, kernel launch overhead, and inter-operator data movement. **Model-level** tools (Paleo, Habitat, AMALI) account for graph-level effects at 5–12% error. **System-level** tools (ASTRA-sim, VIDUR, SimAI) capture communication and scheduling at 5–15% error, with kernel-level errors propagating through the composition chain. Figure 2 illustrates this hierarchy.

4.3 Workload Coverage

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

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.	DNN 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

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	✓	○	—	—	—

The table reveals that **no surveyed tool has been validated on diffusion models or dynamic inference workloads [36]**, only Frontier [18] has validated MoE support, and no single tool offers 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 employ specialized dataflows and memory hierarchies optimized for tensor operations [63], and their computational regularity makes this domain particularly amenable to analytical modeling.

Analytical frameworks dominate. Timeloop [48] computes data reuse, latency, and energy from loop-nest representations at 5–10% error versus RTL simulation with 2000× speedup, and provides deterministic reference outputs for standard designs (Eyeriss [11], Simba). MAESTRO [38] simplifies specification via data-centric directives but is less precise for energy modeling. Sparseloop [69] extends Timeloop to sparse tensors by modeling sparsity patterns, compression formats (CSR, bitmap), and hardware intersection units—critical for transformer inference but limited to static, known sparsity distributions.

Simulation and ML-augmented approaches. PyTorchSim [35] integrates PyTorch 2 with cycle-accurate NPU simulation, directly consuming computation graphs to eliminate manual workload translation, but does not report real-hardware accuracy and inherits simulation speed limitations. ArchGym [37] connects ML surrogates to analytical simulators for design space exploration; its 0.61% RMSE measures surrogate-vs-simulator fidelity, not real-hardware accuracy. **Synthesis.** Accelerator modeling is the most mature subdomain, with Timeloop achieving a favorable accuracy-speed-interpretability balance as the de facto DSE standard. The progression from Timeloop through Sparseloop to PIM-aware tools illustrates a recurring pattern: each extension addresses a new workload characteristic but erodes the simplicity advantage of analytical approaches. The key gap is silicon validation—neither ArchGym nor PyTorchSim validates against manufactured hardware, leaving all accelerator tools anchored to RTL comparisons. Emerging PIM modeling tools [23, 28, 40, 49] target attention acceleration and PIM-GPU co-simulation but lack real PIM hardware validation (Section 8).

5.2 GPU Performance Modeling

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

Cycle-accurate simulation. GPGPU-Sim [4] and Accel-Sim [34] achieve 0.90–0.97 IPC correlation; recent reverse-engineering of modern GPU cores [27] improved Accel-Sim to 13.98% MAPE. However, 1000–10000× slowdown makes these tools impractical at production scale.

Analytical models. The roofline model [67] provides upper bounds but misses dynamic effects; Roofline-LLM [29] extends it to LLM inference. AMALI [9] reduces LLM inference MAPE from 127% to 23.6% via memory hierarchy modeling—the residual error

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
<i>DNN Accelerator Modeling</i>						
Timeloop [48]	NPU	A	Latency/Energy	5–10%	μs	Loop-nest DSE
MAESTRO [38]	NPU	A	Latency/Energy	5–15%	μs	Data-centric directives
Sparseloop [69]	NPU	A	Sparse tensors	5–10%	μs	Compression modeling
PyTorchSim [35]	NPU	S	Cycle-accurate	N/A [‡]	Hours	PyTorch 2 integration
ArchGym [37]	Multi	H	Multi-objective	0.61%*	ms	ML-aided DSE
<i>GPU Performance Modeling</i>						
Accel-Sim [34]	GPU	S	Cycle-accurate	10–20%	Hours	SASS trace-driven
GPGPU-Sim [4]	GPU	S	Cycle-accurate	10–20%	Hours	CUDA workloads
AMALI [9]	GPU	A	LLM inference	23.6%	ms	Memory hierarchy
NeuSight [42]	GPU	H	Kernel/E2E latency	2.3%	ms	Tile-based prediction
Habitat [72]	GPU	H	Training time	11.8%	Per-kernel	Wave scaling
<i>Distributed Training and LLM Serving</i>						
ASTRA-sim [68]	Distributed	T	Training time	5–15%	Minutes	Collective modeling
SimAI [65]	Distributed	T	Training time	1.9%	Minutes	Full-stack simulation
Lumos [44]	Distributed	T	LLM training	3.3%	Minutes	H100 training
VIDUR [3]	GPU cluster	T	LLM serving	<5%	Seconds	Prefill/decode phases
Frontier [18]	Distributed	T	MoE inference	—	Minutes	Stage-centric sim.
TrioSim [43]	Multi-GPU	T	DNN training	N/A [‡]	Minutes	Lightweight multi-GPU
<i>Edge Device Modeling</i>						
nn-Meter [74]	Edge	M	Latency	<1% [†]	ms	Kernel detection
LitePred [16]	Edge	M	Latency	0.7%	ms	85-platform transfer
HELP [41]	Multi	M	Latency	1.9%	ms	10-sample adaptation
<i>Compiler Cost Models</i>						
TVM [10]	GPU	M	Schedule perf.	~15%	ms	Autotuning guidance
Ansor [75]	GPU	M	Schedule perf.	~15%	ms	Program sampling
TLP [73]	GPU	M	Tensor program	<10%	ms	Transformer cost model

reflects the fundamental difficulty of analytically capturing GPU dynamic behavior (warp scheduling, cache contention).

Hybrid learned models. NeuSight [42] achieves 2.3% MAPE on GPT-3 inference (H100/A100/V100) via tile-based prediction mirroring CUDA execution. Habitat [72] uses wave scaling to decompose compute and memory components, achieving 11.8% cross-GPU transfer error (e.g., V100→A100), but requires source GPU profiling and assumes occupancy patterns remain similar across generations. Direct comparison requires caution: NeuSight targets 2023–2025 hardware with LLMs, while Habitat was designed for earlier GPUs with CNNs.

LLM-specific modeling. LLM execution exhibits distinct pre-fill (compute-bound) and decode (memory-bound) phases [51, 77]. VIDUR [3] simulates LLM serving with scheduling strategies (Orca [71], Sarathi [2]) at <5% error. LIFE [17] provides hardware-agnostic analytical inference modeling; HERMES [5] targets heterogeneous disaggregated serving; and emerging predictors include Omniwise [21] and SwizzlePerf [64].

Compiler cost models. TVM [10] and Ansor [75] use ML cost models for autotuning at ~15% MAPE, with TenSet [76] enabling pre-training across diverse programs. TLP [73] uses transformer-based modeling for irregular workloads at <10% MAPE. SynPerf [66]

uses performance models to guide kernel synthesis; both SwizzlePerf [64] and SynPerf prioritize ranking accuracy over absolute error.

Synthesis. GPU modeling exhibits the widest methodological spread: cycle-accurate simulation, analytical models, hybrid learned approaches, and compiler cost models all target the same hardware with error rates spanning 2%–24%. This diversity reflects the fundamental tension between microarchitectural complexity (making analytical modeling hard) and rapid hardware evolution (invalidating training data for learned approaches). NeuSight currently offers the best accuracy–speed trade-off for LLM workloads, but per-GPU profiling limits pre-silicon use—a gap that AMALI and LIFE fill despite higher error. The compiler cost model ecosystem (TVM, TLP, SynPerf) represents a distinct use case where relative ranking matters more than absolute prediction.

5.3 Distributed Training and LLM Serving

Distributed systems introduce communication overhead, synchronization barriers, and parallelism strategy choices across tensor [59], pipeline [26], and data parallelism with memory-efficient optimizers [53].

Training simulation. ASTRA-sim [68] provides end-to-end training simulation via Chakra traces [60] at 5–15% error versus HGX-H100 clusters. SimAI [65] achieves 1.9% MAPE at Alibaba Cloud scale; Lumos [44] achieves 3.3% error on H100 training; Echo [7] and TrioSim [43] offer additional training simulation approaches. PRISM [19] produces prediction intervals at 10K+ GPU scale, capturing stochastic variation.

Scaling and parallelism. Paleo [52] pioneered analytical training-time estimation; MAD Max [25] extends this per parallelism dimension; Sailor [61] addresses heterogeneous clusters. The Llama 3 study [13] documents 4D parallelism at 16K H100 GPUs as validation ground truth.

Inference serving. VIDUR [3] extends to distributed serving with vLLM [39] scheduling; Frontier [18] targets MoE and disaggregated inference; ThrottLL'eM [32] models GPU power management effects on inference. KV cache management is the key serving bottleneck (vLLM's PagedAttention [39] achieves 2–4× throughput), modeled by VIDUR at the system level. Algorithmic innovations like speculative decoding [8] create a moving-target challenge for all serving simulators.

Synthesis. Distributed modeling is the fastest-growing subdomain, bifurcating into *trace-driven fidelity* (ASTRA-sim, SimAI) and *analytical decomposition* (Paleo, MAD Max), with PRISM's probabilistic approach as an emerging third path.

5.4 Edge Device Modeling

Edge devices impose strict power, memory, and latency constraints, and their hardware diversity (mobile CPUs, GPUs, NPUs, DSPs) makes per-device analytical modeling impractical, driving ML-augmented approaches.

nn-Meter [74] reports <1% MAPE using random forest ensembles, but this claim is unverifiable: pre-trained predictors fail with modern scikit-learn versions, scoring 3/10 in our reproducibility evaluation (Section 7). LitePred [16] achieves 0.7% MAPE across 85 platforms using VAE-based intelligent sampling that reduces adaptation to under one hour per device, though the platforms are predominantly ARM-based and transfer to NPUs/DSPs likely degrades. HELP [41] achieves 1.9% MAPE with MAML-style 10-sample adaptation, though accuracy depends on selecting representative operators for the target workload. ESM [46] systematically evaluates surrogate models for hardware-aware NAS, finding that well-tuned random forests match deep learning surrogates—suggesting the field may over-invest in model complexity relative to data quality. Transfer learning provides 22.5% average improvement, up to 87.6% on challenging cross-platform transfers [15].

Synthesis. Edge modeling is dominated by ML-augmented approaches, with the central challenge being *generalization* to new devices with minimal profiling. ESM's finding that simple models match complex ones, combined with nn-Meter's reproducibility failure, suggests that data quality and tool longevity matter more than model sophistication.

5.5 Cross-Cutting Themes

Several themes cut across platform categories. *Structural decomposition* mirroring hardware execution outperforms black-box approaches: Timeloop's loop nests, NeuSight's tiles, and VIDUR's

prefill/decode separation all achieve strong accuracy by encoding domain knowledge into the model structure, while modular pluggable backends (ASTRA-sim, VIDUR) enable broad adoption. *Verifiable moderate accuracy* matters more than *unverifiable high accuracy*—Timeloop (9/10 reproducibility) and ASTRA-sim (8.5/10) see sustained adoption, while nn-Meter (<1% MAPE claimed, 3/10 reproducibility) sees decline.

A persistent **accuracy–generality–speed trilemma** explains why no single methodology has displaced others: cycle-accurate simulators maximize accuracy but sacrifice speed; analytical models maximize speed but sacrifice accuracy on complex hardware; ML-augmented approaches achieve both but sacrifice generality, requiring retraining when hardware changes. The maturity of each subdomain mirrors economic incentive: accelerator DSE is most mature (irreversible chip design errors), distributed training simulation is fastest-growing (million-dollar training runs), and edge modeling has weakest reproducibility (lower deployment costs).

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.

6.1 Accuracy by Problem Difficulty

We organize accuracy results by inherent problem difficulty rather than comparing across incompatible benchmarks (Figure 3). Accelerator dataflow modeling is most tractable (Timeloop: 5–10%); single-GPU kernel prediction achieves 2–12% via hybrid methods (NeuSight, Habitat); distributed systems reach 2–15% (SimAI 1.9%, ASTRA-sim 5–15%); cross-platform edge prediction achieves 0.7–2% but requires per-device profiling; and GPU analytical modeling remains hardest (AMALI: 23.6%). Setup costs vary dramatically: analytical models require only architecture specifications, 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. 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 prioritize tools with Docker-first deployment (VIDUR, Timeloop, ASTRA-sim) over tools with unpinned dependencies, as our evaluation shows reproducibility scores strongly predict long-term usability.

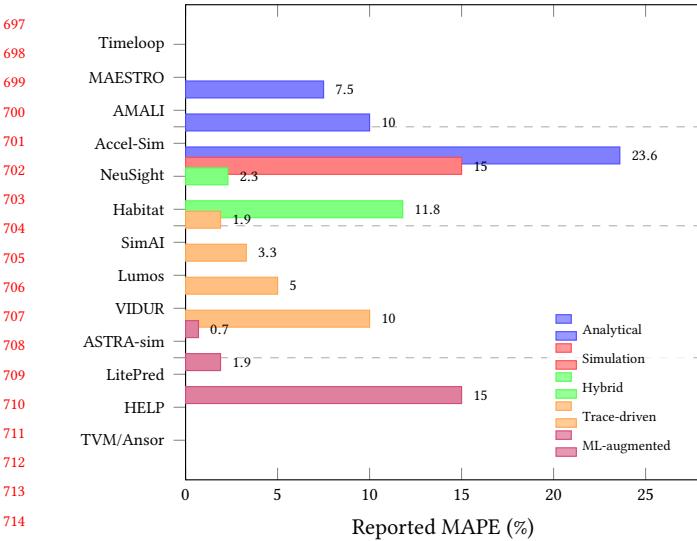


Figure 3: Reported accuracy (MAPE) of surveyed tools, grouped by methodology type. Range midpoints used where ranges are reported. Cross-tool comparison is approximate due to differing benchmarks, workloads, and hardware targets.

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

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.

7.1 Per-Tool Results

VIDUR (9/10)—the highest-scoring tool. We simulated Llama-2-7B inference on a simulated A100 using vLLM and Sarathi scheduling (Table 5). VIDUR correctly captures scheduling trade-offs: Sarathi achieves 12.2% lower latency than vLLM (0.158 s vs. 0.177 s) via chunked prefill [2], TPOT differs by only 3.5% (confirming

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%

hardware-bound decode), and vLLM preempted 26.5% of requests while Sarathi preempted zero—matching the algorithmic difference in KV-cache management [39]. VIDUR’s Docker-pinned dependencies avoid the serialization failures seen in nn-Meter.

Timeloop (9/10). The Docker image provides CLI tools that work correctly for Eyeriss-like configurations with fully deterministic, bit-identical outputs—a significant reproducibility strength. Reference outputs for standard designs (Eyeriss, Simba) enable verification without hardware. However, Python bindings fail (`ImportError: libbarvinok.so.23`), preventing programmatic use.

ASTRA-sim (8.5/10). We executed 8-NPU collective microbenchmarks and ResNet-50 data-parallel training at 2–8 GPUs on HGX-H100 (Table 6). Collective operation ratios are physically plausible: Reduce-Scatter takes half the time of All-Reduce (consistent with half the data), while All-to-All takes ~2× All-Reduce. Communication overhead scales from 0.05% (2 GPUs) to 0.30% (8 GPUs)—a 5.76× increase for 4× more GPUs, consistent with ring All-Reduce scaling. Scale coverage is limited to 8 NPUs by included topology files.

NeuSight (7.5/10). The tile-based decomposition correctly mirrors CUDA tiling for standard dense operations, but testing on irregular workloads was limited by missing examples, suggesting the tool is best validated for regular LLM workloads.

nn-Meter (3/10)—the lowest-scoring tool. After four installation attempts (>4 hours), we could not execute *any* predictions due to a

chain of unpinned dependencies: pickle-serialized predictors (scikit-learn 0.23.1) are incompatible with current versions, and onnx-simplifier fails on aarch64. The claimed <1% MAPE is **unverifiable on any current software stack**; the tool has received no updates since 2022.

7.2 Lessons and Threats to Validity

Five lessons emerge: (1) **Docker-first deployment** is the strongest reproducibility predictor (Docker tools: 8.5+/10; nn-Meter without Docker: 3/10). (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 [13]. (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. Three dimensions of generalization remain largely unsolved. *Workload generalization*: nearly all accuracy numbers are measured on CNNs, with CNN→transformer transfer largely unvalidated (NeuSight is a notable exception); MoE, diffusion models, and dynamic inference [36] have almost no validated prediction tools. Neural scaling laws [12, 20, 24, 33] predict training loss but not hardware-specific latency. Figure 4 shows the shift from CNN-only validation toward LLM workloads since 2023, but MoE and diffusion remain uncharacterized. *Hardware generalization*: meta-learning (HELP: 10-sample adaptation), feature-based transfer (LitePred: 85 devices), and analytical decomposition (Habitat) show promise, but cross-family transfer (GPU→TPU→PIM) remains unsolved. *Temporal generalization*—software stack evolution silently invalidating trained models—is addressed by no surveyed tool.

The composition problem. Composing kernel-level predictions into end-to-end estimates is unsolved: NeuSight’s 2.3% kernel MAPE yields $\sim 10\times$ 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 architectures [23, 28, 40, 49], chiplets, and disaggregated designs blur conventional memory hierarchy assumptions; hardware-aware optimizations (FlashAttention [14]) change the performance landscape faster than models can be retrained. On the reproducibility front, no equivalent of MLPerf [45, 57] exists for performance *prediction*—standardized prediction benchmarks would significantly advance the field. Analytical models remain the most interpretable: Timeloop identifies data movement bottlenecks, MAESTRO reveals suboptimal dataflows, and VIDUR exposes scheduling inefficiencies, while ML-augmented approaches offer limited causal understanding.

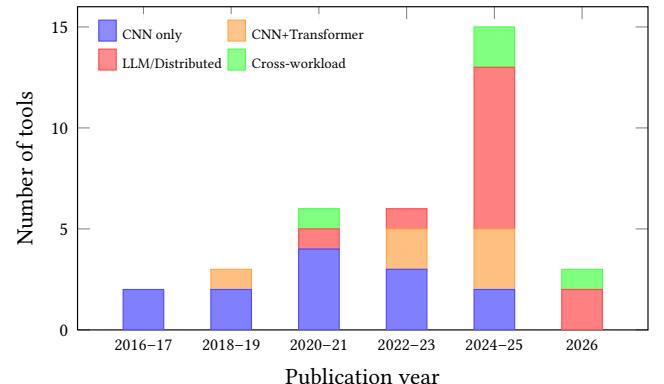


Figure 4: Workload coverage of surveyed tools by publication period. The shift toward transformer and LLM workloads accelerates from 2023, but MoE and diffusion models remain largely uncharacterized.

Future directions. Five priorities: (1) transformer/MoE-aware tools with validated non-CNN accuracy; (2) validated composition methods with bounded end-to-end error; (3) unified energy-latency-memory prediction [56]; (4) temporal robustness benchmarks for software stack evolution; (5) unified tooling with Docker-first deployment, portable formats (ONNX), and standard workload representations (Chakra [60]).

9 Conclusion

This survey analyzed approximately 25 tools for predicting ML workload performance, organized by methodology type, target platform, and abstraction level. Key findings: (1) *Methodology determines trade-offs, not quality*—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). (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 Shoib, 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/10.1145/204716.2024718>
- [7] 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).
- [8] 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.
- [9] 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.
- [10] 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.
- [11] 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>
- [12] 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.
- [13] Weiwei Chu, Xinfeng Xie, Jiecas Yu, Jie Wang, Pavav 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] Paraskevas Gavriliidis 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] Amer 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.
- [22] 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.
- [23] 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 Inference. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPOLOS)*. 1–17. NPU-PIM heterogeneous architecture for LLM inference with performance modeling. KAIST/Georgia Tech.
- [24] 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.
- [25] 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>
- [26] 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.
- [27] Rodrigo Huerta, Mojtaba Abae 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.
- [28] 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.
- [29] 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).
- [30] 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.
- [31] 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.
- [32] Andreas Kosmas Kakolyris, Dimosthenis Masouros, Petros Vavaroutsos, Sotirios Kydis, and Dimitrios Soudris. 2025. throttLLM: 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.
- [33] 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.
- [34] 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>
- [35] 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.
- [36] 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.
- [37] Srivatsan Krishnan, Amir Yazdanbakhsh, Shvetank Prakash, Norman P. Jouppi, Jignesh Parmar, Hyoukjun Kim, James Laudon, and Chandrakanth 1044

- 1045 Narayanaswami. 2023. ArchGym: An Open-Source Gymnasium for Machine
1046 Learning Assisted Architecture Design. In *Proceedings of the 50th International*
1047 *Symposium on Computer Architecture (ISCA)*. 1–16. <https://doi.org/10.1145/3579371.3589049>
- 1048 [38] Hyoukjun Kwon, Prasanth Chatarasi, Michael Barber, Michael Pellauer, Angshuman
1049 Parashar, and Tushar Krishna. 2019. MAESTRO: A Data-Centric Approach
1050 to Understand Reuse, Performance, and Hardware Cost of DNN Mappings. In
1051 *Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*.
1052 1–14. <https://doi.org/10.1145/3352460.3358292>
- 1053 [39] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng,
1054 Cody Hao Yu, Joseph E. Gonzalez, Hau Zhang, and Ion Stoica. 2023. Efficient
1055 Memory Management for Large Language Model Serving with PagedAttention.
1056 In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.
1057 611–626. <https://doi.org/10.1145/3600006.3613165>
- 1058 [40] Hyojung Lee, Daehyeon Baek, Jimyoung Son, Jeun Choi, Kihyo Moon, and
1059 Minsung Jang. 2025. PAISE: PIM-Accelerated Inference Scheduling Engine for
1060 Transformer-based LLM. In *Proceedings of the IEEE International Symposium on High*
1061 *Performance Computer Architecture (HPCA)*. 1–14. PIM-based LLM
1062 inference scheduling, 48.3% speedup, 11.5% power reduction. Samsung.
- 1063 [41] Hayeon Lee, Sewoong Lee, Song Chong, and Sung Ju Hwang. 2021. HELP:
1064 Hardware-Adaptive Efficient Latency Prediction for NAS via Meta-Learning.
1065 In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 34. 27016–
1066 27028.
- 1067 [42] Seunghyun Lee, Amar Phanishayee, and Divya Mahajan. 2025. NeuSight: GPU
1068 Performance Forecasting via Tile-Based Execution Analysis. In *Proceedings of the 30th ACM*
1069 *International Conference on Architectural Support for Programming Languages and*
1070 *Operating Systems (ASPLOS)*. 1–15.
- 1071 [43] Jianbo Li et al. 2025. TrioSim: A Lightweight Simulator for Large-Scale DNN
1072 Workloads on Multi-GPU Systems. In *Proceedings of the 52nd Annual International*
1073 *Symposium on Computer Architecture (ISCA)*. 1–13. Multi-GPU DNN simulation
1074 with lightweight approach for distributed training analysis.
- 1075 [44] Wenzuan Liang et al. 2025. Lumos: Efficient Performance Modeling and Estimation
1076 for Large-scale LLM Training. In *Proceedings of Machine Learning and*
1077 *Systems (MLSys)*. 1–16. Trace-driven performance modeling achieving 3.3% error
1078 on H100 GPUs for LLM training.
- 1079 [45] Peter Mattson, Christine Cheng, Cody Coleman, Greg Diamos, Paulius Micikevicius,
1080 David Patterson, Hanlin Tang, Gu-Yeon Wei, Peter Bailis, Victor Bittorf, et al. 2020. MLPerf
1081 Training Benchmark. In *Proceedings of Machine Learning and*
1082 *Systems (MLSys)*. 336–349. Standard ML training benchmark suite covering
1083 image classification, object detection, NLP, recommendation, reinforcement
1084 learning.
- 1085 [46] Azaz-Ur-Rehman Nasir, Samroz Ahmad Shoaib, Muhammad Abdullah Hanif, and
1086 Muhammad Shafique. 2025. ESM: A Framework for Building Effective Surrogate
1087 Models for Hardware-Aware Neural Architecture Search. In *Proceedings of the*
1088 *62nd ACM/IEEE Design Automation Conference (DAC)*. 1–6. 97.6% accuracy
1089 surrogate model framework for HW-aware NAS.
- 1090 [47] Amir Nasr-Esfahany et al. 2025. Concorde: Fast and Accurate CPU Performance
1091 Modeling with Compositional Analytical-ML Fusion. In *Proceedings of the 52nd*
1092 *Annual International Symposium on Computer Architecture (ISCA)*. 1–15. Hybrid
1093 analytical-ML approach achieving 2% CPI error at 5 orders of magnitude faster
1094 than gem5.
- 1095 [48] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen,
1096 Victor A. Ying, Anurag Muber, Rangharajan Venkatesan, Brucek Khalany,
1097 Stephen W. Keckler, and Joel Emer. 2019. Timeloop: A Systematic Approach
1098 to DNN Accelerator Evaluation. In *Proceedings of the IEEE International*
1099 *Symposium on Performance Analysis of Systems and Software (ISPASS)*. 304–315.
1100 <https://doi.org/10.1109/ISPASS2019.00042>
- 1101 [49] Jaehyun Park, Jaewan Choi, Kwanhee Kyung, Michael Jaemi Kim, Yongsuk
1102 Kwon, Nam Sung Kim, and Jung Ho Ahn. 2024. AttAcc! Unleashing the Power of
1103 PIM for Batched Transformer-based Generative Model Inference. In *Proceedings of*
1104 *the 29th ACM International Conference on Architectural Support for Programming*
1105 *Languages and Operating Systems (ASPLOS)*. 1–16. PIM-based accelerator for
1106 batched transformer attention. Seoul National University/UTUC..
- 1107 [50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory
1108 Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019.
1109 PyTorch: An Imperative Style, High-Performance Deep Learning Library. In
1110 *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 32. 8024–8035.
- 1111 [51] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aakanksha Shah, Íñigo Goiri,
1112 Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM
1113 Inference Using Phase Splitting. In *Proceedings of the 51st Annual International*
1114 *Symposium on Computer Architecture (ISCA)*. 118–132. <https://doi.org/10.1109/ISCA59077.2024.00019> Best Paper Award.
- 1115 [52] Hang Qi, Evan R. Sparks, and Ameet Talwalkar. 2017. Paleo: A Performance
1116 Model for Deep Neural Networks. In *Proceedings of the 5th International Conference*
1117 *on Learning Representations (ICLR)*. <https://openreview.net/forum?id=SyVVJ85lg>
- 1118 [53] Samyam Rajbhandari, Jeff Rasley, Olatunji Rber, and Yuxiong He. 2020. ZERO:
1119 Memory Optimizations Toward Training Trillion Parameter Models. In *Proceedings*
1120 *of the International Conference on High Performance Computing, Networking,*
1121 *Storage and Analysis (SC)*. 1–16. <https://doi.org/10.1109/SC41405.2020.00024>
- 1122 DeepSpeed ZeRO optimizer partitioning for memory-efficient distributed training.
- 1123 [54] Mehdi Rakhshanfar and Aliakbar Zarandi. 2021. A Survey on Machine Learning-
1124 Based Design Space Exploration for Processor Architectures. *Journal of Systems*
1125 *Architecture* 121 (2021), 102339. <https://doi.org/10.1016/j.sysarc.2021.102339>
- 1126 [55] Saeed Rashidi, Srinivas Srinivasan, Kazem Hamedani, and Tushar Krishna.
1127 2020. ASTRA-SIM: Enabling SW/HW Co-Design Exploration for Distributed
1128 DL Training Platforms. In *Proceedings of the IEEE International Symposium*
1129 *on Performance Analysis of Systems and Software (ISPASS)*. 81–92. <https://doi.org/10.1109/ISPASS48437.2020.00018>
- 1130 [56] Vijay Janapa Reddi et al. 2025. MLPerf Power: Benchmarking the Energy Efficiency
1131 of Machine Learning Inference. In *Proceedings of the IEEE International*
1132 *Symposium on High Performance Computer Architecture (HPCA)*. 1–14. Energy
1133 efficiency benchmarking for ML inference workloads.
- 1134 [57] Vijay Janapa Reddi, Christine Cheng, David Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, Maxim Breeshevskov, Mark Duber, et al. 2020. MLPerf Inference Benchmark. In *Proceedings of the 47th International*
1135 *Symposium on Computer Architecture (ISCA)*. 446–459. <https://doi.org/10.1109/ISCA45697.2020.00045> Standard ML inference benchmark suite with server and
1136 offline scenarios.
- 1137 [58] Timothy Sherwood, Erez Perelman, Greg Hamerly, and Brad Calder. 2002. Automatically
1138 Characterizing Large Scale Program Behavior. In *Proceedings of the 10th International*
1139 *Conference on Architectural Support for Programming Languages and*
1140 *Operating Systems (ASPLOS)*. 45–57. <https://doi.org/10.1145/605397.605403>
- 1141 Introduces SimPoint: automatic selection of representative simulation points
1142 using k-means clustering.
- 1143 [59] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper,
1144 and Bryan Catanzaro. 2020. Megatron-LM: Training Multi-Billion Parameter
1145 Language Models Using Model Parallelism. In *ArXiv preprint arXiv:1909.08053*.
1146 Intra-layer tensor parallelism for large language model training.
- 1147 [60] Srinivas Sridharan, Taekyung Heo, Jinwoo Choi, Garyfallia Yu, Saeed Rashidi,
1148 William Won, Zhaodong Meng, and Tushar Krishna. 2023. Chakra: Advancing
1149 Performance Benchmarking and Co-design using Standardized Execution Traces.
1150 *arXiv preprint arXiv:2305.14516* (2023).
- 1151 [61] Foteini Strati, Zhendong Zhang, George Manos, Ixeia Sanchez Periz, Qinghao
1152 Hu, Tiancheng Chen, Berk Buzcu, Song Han, Pamela Delgado, and Ana Klimovic.
1153 2025. Sailor: Automating Distributed Training over Dynamic, Heterogeneous,
1154 and Geo-distributed Clusters. In *Proceedings of the 30th ACM Symposium on*
1155 *Operating Systems Principles (SOSP)*. 1–18. Automated distributed training with
1156 runtime/memory simulation over heterogeneous resources. ETH Zurich/MIT.
- 1157 [62] Ondrej Sykora, Alexis Rucker, Charith Mendis, Rajkishore Barik, Phitchaya
1158 Mangpo Phothilmatha, and Saman Amarasinghe. 2022. GRANITE: A Graph Neural
1159 Network Model for Basic Block Throughput Estimation. In *Proceedings of the IEEE*
1160 *International Symposium on Workload Characterization (ISWC)*. 1–13. <https://doi.org/10.1109/ISWC55918.2022.00014>
- 1161 [63] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. 2017. Efficient
1162 Processing of Deep Neural Networks: A Tutorial and Survey. In *Proceedings*
1163 *of the IEEE*, Vol. 105. 2295–2329. <https://doi.org/10.1109/JPROC.2017.2761740>
- 1164 Canonical DNN accelerator taxonomy covering dataflows, data reuse, and energy
1165 efficiency.
- 1166 [64] Adrian Tschandl, Mohamed Awad, et al. 2025. SwizzlePerf: Hardware-Aware
1167 LLMs for GPU Kernel Performance Optimization. *arXiv preprint arXiv:2508.20258*
1168 (2025). LLM-based spatial optimization for GPU kernels, up to 2.06x speedup
1169 via swizzling.
- 1170 [65] Xizheng Wang, Qingxu Li, Yichi Xu, Gang Lu, Heyang Zhou, Sen Zhang, Yikai
1171 Zhu, Yang Liu, Pengcheng Zhang, Kun Qian, et al. 2025. SimAI: Unifying Ar-
1172 chitecture Design and Performance Tuning for Large-Scale LLM Training with
1173 Scalability and Precision. In *Proceedings of the 22nd USENIX Symposium on*
1174 *Networked Systems Design and Implementation (NSDI)*. 1–18. Full-stack LLM
1175 training simulator achieving 98.1% alignment with real-world results. Alibaba
1176 Cloud/Tsinghua.
- 1177 [66] Zixian Wang et al. 2025. SynPerf: Synthesizing High-Performance GPU Kernels
1178 via Pipeline Decomposition. *arXiv preprint* (2025). Under review.
- 1179 [67] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An
1180 Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (2009), 65–76. <https://doi.org/10.1145/1498765.1498785>
- 1181 [68] William Won, Taekyung Heo, Saeed Rashidi, Saeed Talati, Srinivas Srinivasan,
1182 and Tushar Krishna. 2023. ASTRA-sim2.0: Modeling Hierarchical Networks and
1183 Disaggregated Systems for Large-Model Training at Scale. In *Proceedings of the*
1184 *IEEE International Symposium on Performance Analysis of Systems and Software*
1185 (ISPASS). 283–294. <https://doi.org/10.1109/ISPASS57527.2023.00035>
- 1186 [69] Yannan Nellie Wu, Joel Emer, and Vivienne Sze. 2022. Sparseloop: An Analytical
1187 Approach to Sparse Tensor Accelerator Modeling. In *Proceedings of the 55th*
1188 *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–15. <https://doi.org/10.1109/MICRO55527.2022.983311>

1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160

- 1161 //doi.org/10.1109/MICRO56248.2022.00078
 1162 [70] Roland E. Wunderlich, Thomas F. Wenisch, Babak Falsafi, and James C. Hoe. 2003.
 1163 SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical
 1164 Sampling. In *Proceedings of the 30th Annual International Symposium on Computer
 1165 Architecture (ISCA)*. 84–97. <https://doi.org/10.1109/ISCA.2003.1206991> Statistical
 1166 sampling achieving 0.64% CPI error with 35x speedup over detailed simulation.
 1167 [71] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-
 1168 Gon Chun. 2022. ORCA: A Distributed Serving System for Transformer-Based
 1169 Generative Models. In *Proceedings of the 16th USENIX Symposium on Operating
 1170 Systems Design and Implementation (OSDI)*. 521–538.
 1171 [72] Geoffrey X. Yu, Yubo Gao, Pavel Golber, and Asaf Cidon. 2021. Habitat: A
 1172 Runtime-Based Computational Performance Predictor for Deep Neural Network
 1173 Training. In *Proceedings of the USENIX Annual Technical Conference (ATC)*. 503–
 1174 521.
 1175 [73] Yi Zhai, Yu Cheng Wang, Peng Jiang, and Congming Kang. 2023. TLP: A Deep
 1176 Learning-based Cost Model for Tensor Program Tuning. In *Proceedings of the
 1177 28th ACM International Conference on Architectural Support for Programming
 1178 Languages and Operating Systems (ASPLOS)*. 833–845. <https://doi.org/10.1145/3575693.3575736>
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997
 1998
 1999
 2000
 2001
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 2024
 2025
 2026
 2027
 2028
 2029
 2030
 2031
 2032
 2033
 2034
 2035
 2036
 2037
 2038
 2039
 2040
 2041
 2042
 2043
 2044
 2045
 2046
 2047
 2048
 2049
 2050
 2051
 2052
 2053
 2054
 2055
 2056
 2057
 2058
 2059
 2060
 2061
 2062
 2063
 2064
 2065
 2066
 2067
 2068
 2069
 2070
 2071
 2072
 2073
 2074
 2075
 2076
 2077
 2078
 2079
 2080
 2081
 2082
 2083
 2084
 2085
 2086
 2087
 2088
 2089
 2090
 2091
 2092
 2093
 2094
 2095
 2096
 2097
 2098
 2099
 2100
 2101
 2102
 2103
 2104
 2105
 2106
 2107
 2108
 2109
 2110
 2111
 2112
 2113
 2114
 2115
 2116
 2117
 2118
 2119
 2120
 2121
 2122
 2123
 2124
 2125
 2126
 2127
 2128
 2129
 2130
 2131
 2132
 2133
 2134
 2135
 2136
 2137
 2138
 2139
 2140
 2141
 2142
 2143
 2144
 2145
 2146
 2147
 2148
 2149
 2150
 2151
 2152
 2153
 2154
 2155
 2156
 2157
 2158
 2159
 2160
 2161
 2162
 2163
 2164
 2165
 2166
 2167
 2168
 2169
 2170
 2171
 2172
 2173
 2174
 2175
 2176
 2177
 2178
 2179
 2180
 2181
 2182
 2183
 2184
 2185
 2186
 2187
 2188
 2189
 2190
 2191
 2192
 2193
 2194
 2195
 2196
 2197
 2198
 2199
 2200
 2201
 2202
 2203
 2204
 2205
 2206
 2207
 2208
 2209
 2210
 2211
 2212
 2213
 2214
 2215
 2216
 2217
 2218
 2219
 2220
 2221
 2222
 2223
 2224
 2225
 2226
 2227
 2228
 2229
 2230
 2231
 2232
 2233
 2234
 2235
 2236
 2237
 2238
 2239
 2240
 2241
 2242
 2243
 2244
 2245
 2246
 2247
 2248
 2249
 2250
 2251
 2252
 2253
 2254
 2255
 2256
 2257
 2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286
 2287
 2288
 2289
 2290
 2291
 2292
 2293
 2294
 2295
 2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304
 2305
 2306
 2307
 2308
 2309
 2310
 2311
 2312
 2313
 2314
 2315
 2316
 2317
 2318
 2319
 2320
 2321
 2322
 2323
 2324
 2325
 2326
 2327
 2328
 2329
 2330
 2331
 2332
 2333
 2334
 2335
 2336
 2337
 2338
 2339
 2340
 2341
 2342
 2343
 2344
 2345
 2346
 2347
 2348
 2349
 2350
 2351
 2352
 2353
 2354
 2355
 2356
 2357
 2358
 2359
 2360
 2361
 2362
 2363
 2364
 2365
 2366
 2367
 2368
 2369
 2370
 2371
 2372
 2373
 2374
 2375
 2376
 2377
 2378
 2379
 2380
 2381
 2382
 2383
 2384
 2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397
 2398
 2399
 2400
 2401
 2402
 2403
 2404
 2405
 2406
 2407
 2408
 2409
 2410
 2411
 2412
 2413
 2414
 2415
 2416
 2417
 2418
 2419
 2420
 2421
 2422
 2423
 2424
 2425
 2426
 2427
 2428
 2429
 2430
 2431
 2432
 2433
 2434
 2435
 2436
 2437
 2438
 2439
 2440
 2441
 2442
 2443
 2444
 2445
 2446
 2447
 2448
 2449
 2450
 2451
 2452
 2453
 2454
 2455
 2456
 2457
 2458
 2459
 2460
 2461
 2462
 2463
 2464
 2465
 2466
 2467
 2468
 2469
 2470
 2471
 2472
 2473
 2474
 2475
 2476
 2477
 2478
 2479
 2480
 2481
 2482
 2483
 2484
 2485
 2486
 2487
 2488
 2489
 2490
 2491
 2492
 2493
 2494
 2495
 2496
 2497
 2498
 2499
 2500
 2501
 2502
 2503
 2504
 2505