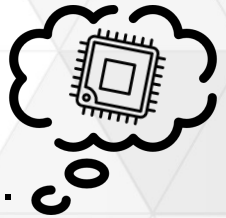
DeepSim: SOL-Based Explainable GPU Performance Projection with Multi-Task Learning

Mengyue Huang[1], Zhanhui Li[1], Tim Purcell[2],
Mingqiao Li[1], Zhengru Wang[1], Nirmal Saxena[2]

[1]NVIDIA, Shanghai, China
[2]NVIDIA, Santa Clara, California, USA



Early-Stage GPU Architecture Design

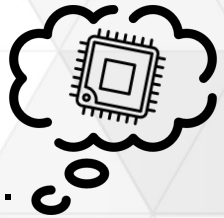


Imagine you are a GPU architect designing a next-generation GPU.

- The target hardware is **not yet available**
- Complete software stacks **have not been finalized**

Goal: *Find the **best architectural knobs** for the target GPU*
e.g., number of SMs, DRAM bandwidth…

Early-Stage GPU Architecture Design

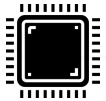


Imagine you are a GPU architect designing a next-generation GPU.

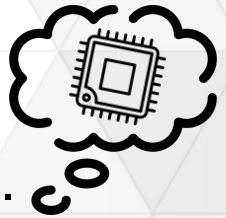
- The target hardware is **not yet available**
- Complete software stacks **have not been finalized**

Goal: *Find the **best architectural knobs** for the target GPU*
e.g., number of SMs, DRAM bandwidth…

✓ What you usually have:

- At least one mature existing GPU 
- Critical real-world GPU kernels 

Early-Stage GPU Architecture Design



Imagine you are a GPU architect designing a next-generation GPU.

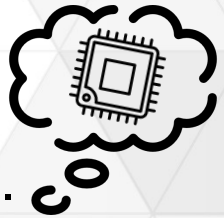
- The target hardware is **not yet available**
- Complete software stacks **have not been finalized**



Goal: *Find the **best architectural knobs** for the target GPU*
e.g., number of SMs, DRAM bandwidth…

Core Problem: How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

Early-Stage GPU Architecture Design



Imagine you are a GPU architect designing a next-generation GPU.

- The target hardware is **not yet available**
- Complete software stacks **have not been finalized**



Goal: *Find the **best architectural knobs** for the target GPU*
e.g., number of SMs, DRAM bandwidth…

Core Problem: How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

Requirements:

- Accurate: Predictions must be reliable
- Fast: Evaluate thousands of knob and kernel combinations
- Explainable: Identify kernel bottlenecks under different knob settings, and reveal potential optimization directions

Existing Methods

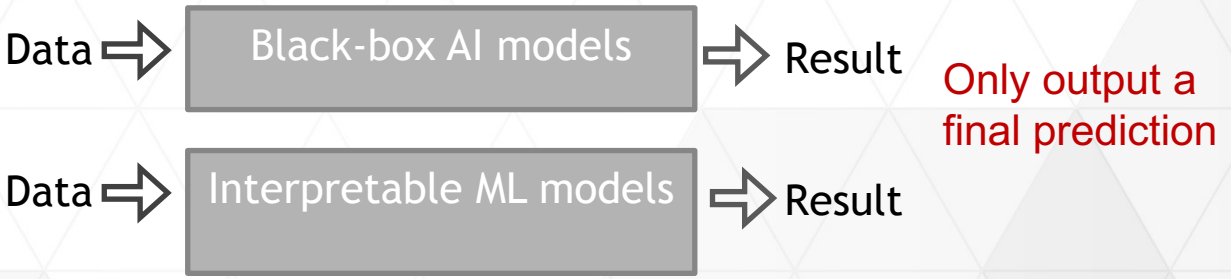
How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

- **Analytical models**: Typically **coarse-grained**, failing to meet target accuracy
- **Simulators**: **Prohibitive runtime** for large-scale design-space exploration
- **Data-driven models**: Often **limited generalization** to unseen kernels; typically **weak explainability**

Existing Methods

How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

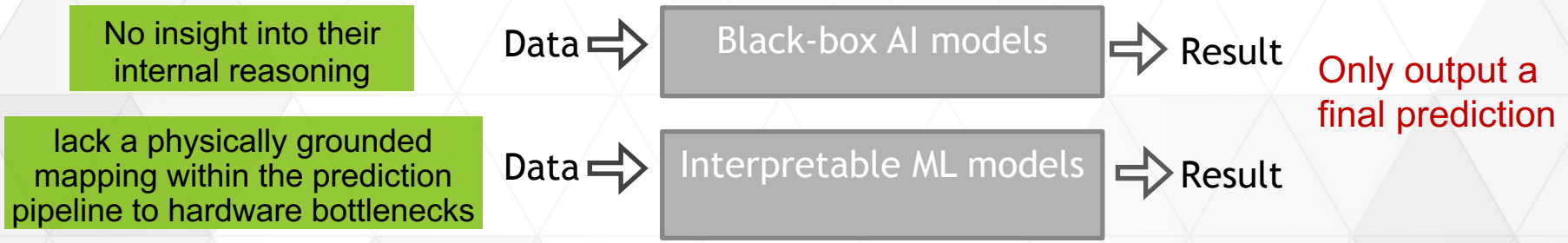
- **Analytical models**: Typically **coarse-grained**, failing to meet target accuracy
- **Simulators**: **Prohibitive runtime** for large-scale design-space exploration
- **Data-driven models**: Often **limited generalization** to unseen kernels; typically **weak explainability**



Existing Methods

How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

- **Analytical models**: Typically **coarse-grained**, failing to meet target accuracy
- **Simulators**: **Prohibitive runtime** for large-scale design-space exploration
- **Data-driven models**: Often **limited generalization** to unseen kernels; typically **weak explainability**



Summary: Existing Methods

How would you predict kernel performance for a target GPU and a given kernel under these early-stage pre-silicon conditions?

Method	Speed	Accuracy	Explainbility
Analytical Model	✓	Insufficient	✓
Simulators	Insufficient	✓	✓
Data-Driven Models	✓	Insufficient	Insufficient

Motivate **DeepSim**: a **fast**, **accurate**, and **explainable** method for **pre-silicon hardware** performance projection.

How to achieve explainability

For the execution of a given kernel on a target GPU

- **SOL ratios** for each hardware unit:
  - Quantify how closely each architectural unit **approaches its idealized throughput limit**
  - Reveals **which unit is the busiest** and where the performance limitation lies

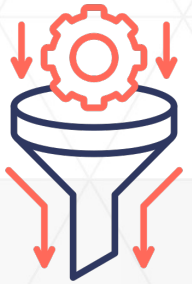


How to achieve explainability

For the execution of a given kernel on a target GPU

- **SOL ratios** for each hardware unit:
  - Quantify how closely each architectural unit **approaches its idealized throughput limit**
  - Reveals **which unit is the busiest** and where the performance limitation lies
- **Fine-grained bottleneck analysis** for this execution:
  - Identify the busiest pipeline part within the constrained unit
  - Pinpoint **the specific limitation dominating overall runtime**

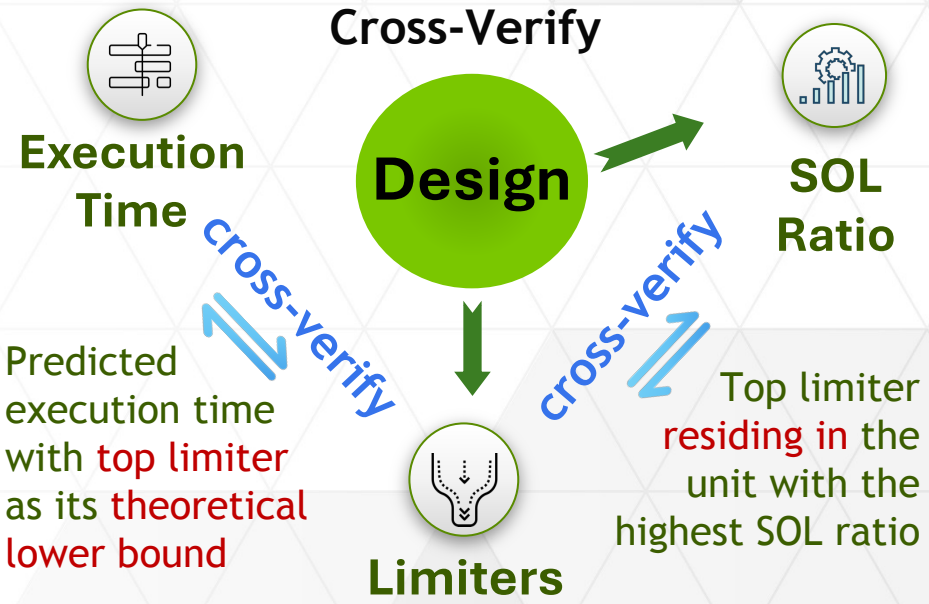




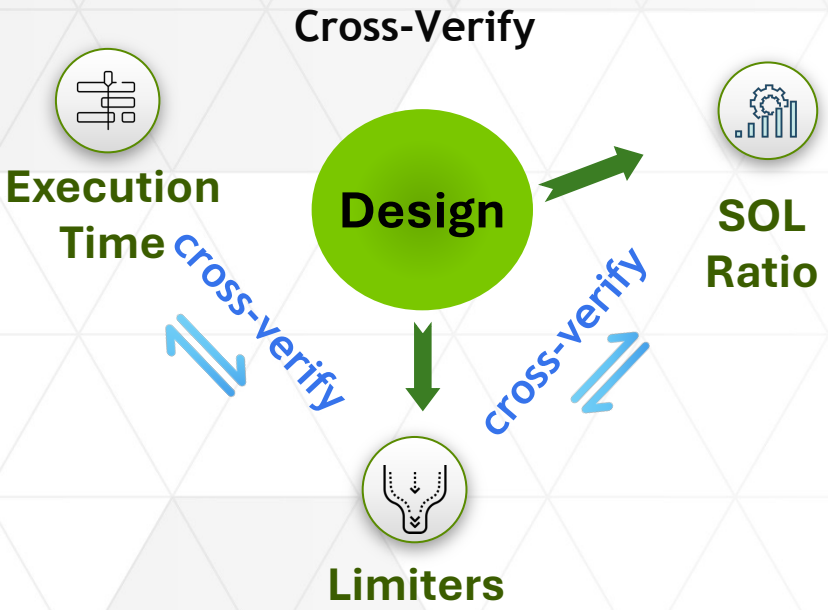
How to achieve explainability

For the execution of a given kernel on a target GPU

- **SOL ratios** for each hardware unit:
  - Quantify how closely each architectural unit **approaches its idealized throughput limit**
  - Reveals **which unit is the busiest** and where the performance limitation lies
- **Fine-grained bottleneck analysis** for this execution:
  - Identify the busiest pipeline part within the constrained unit
  - Pinpoint **the specific limitation dominating overall runtime**



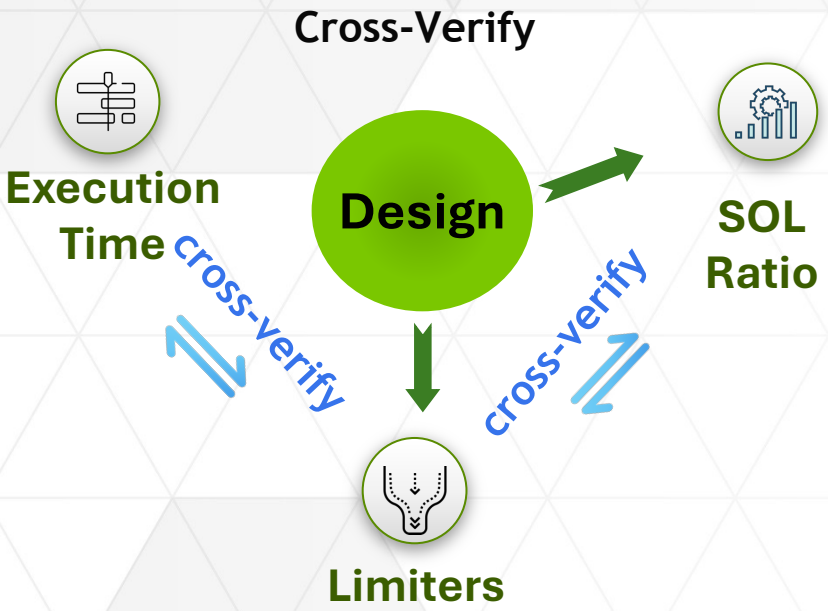
Design: Considerations





- How to estimate SOL ratios on target GPU
- How to derive (top) limiters on target GPU
- How to compute final execution time

Design: Considerations



- How to derive (top) limiters on target GPU
  - **Fine-grained Limiter Modeling** → ideal limiters
  - SOL Correction → practical limiters
- How to estimate SOL ratios on target GPU
  - ML Based **SOL Ratio Prediction**
- How to compute final execution time
  - **Limiter-Driven Runtime Generation**

Design: Fine-grained Limiter Modeling

FTIM (Fine-Grained Traffic-Based and Imbalance-Aware Modeling)

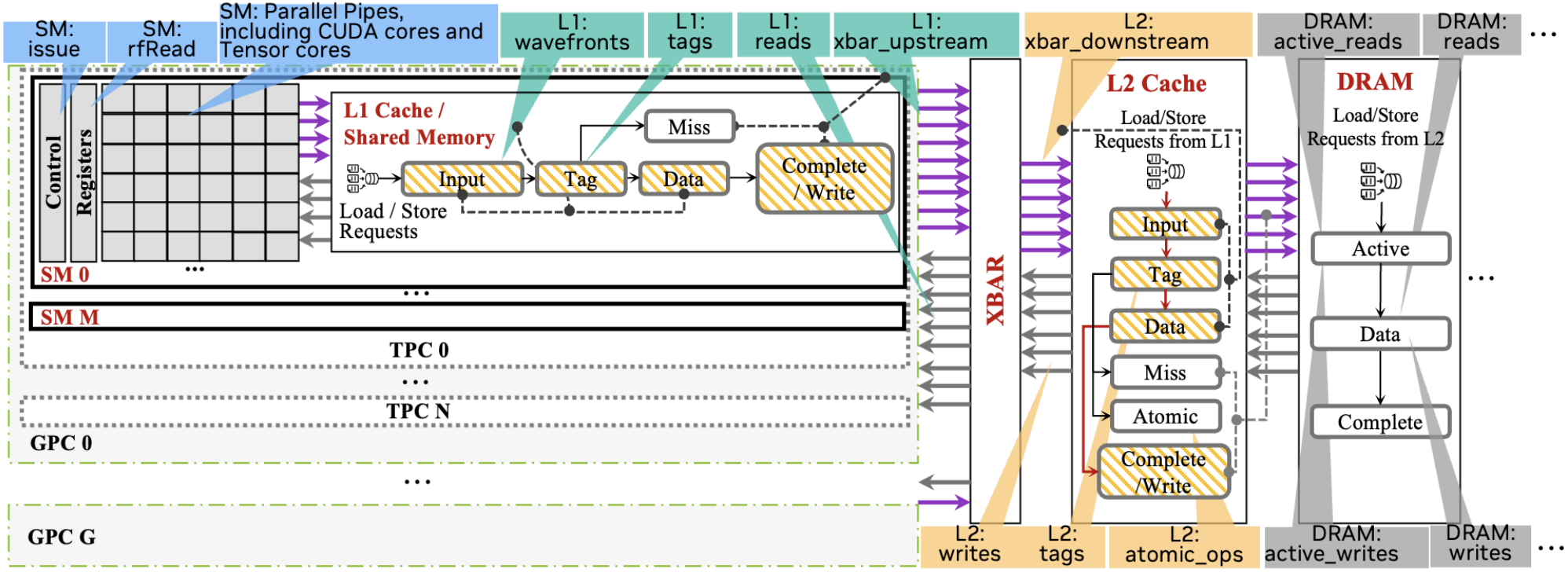
Based on traffic from existing chips

Design: Fine-grained Limiter Modeling

FTIM (Fine-Grained Traffic-Based and Imbalance-Aware Modeling)

Based on traffic from existing chips

- **Fine-Grained**: intra-unit pipeline analysis

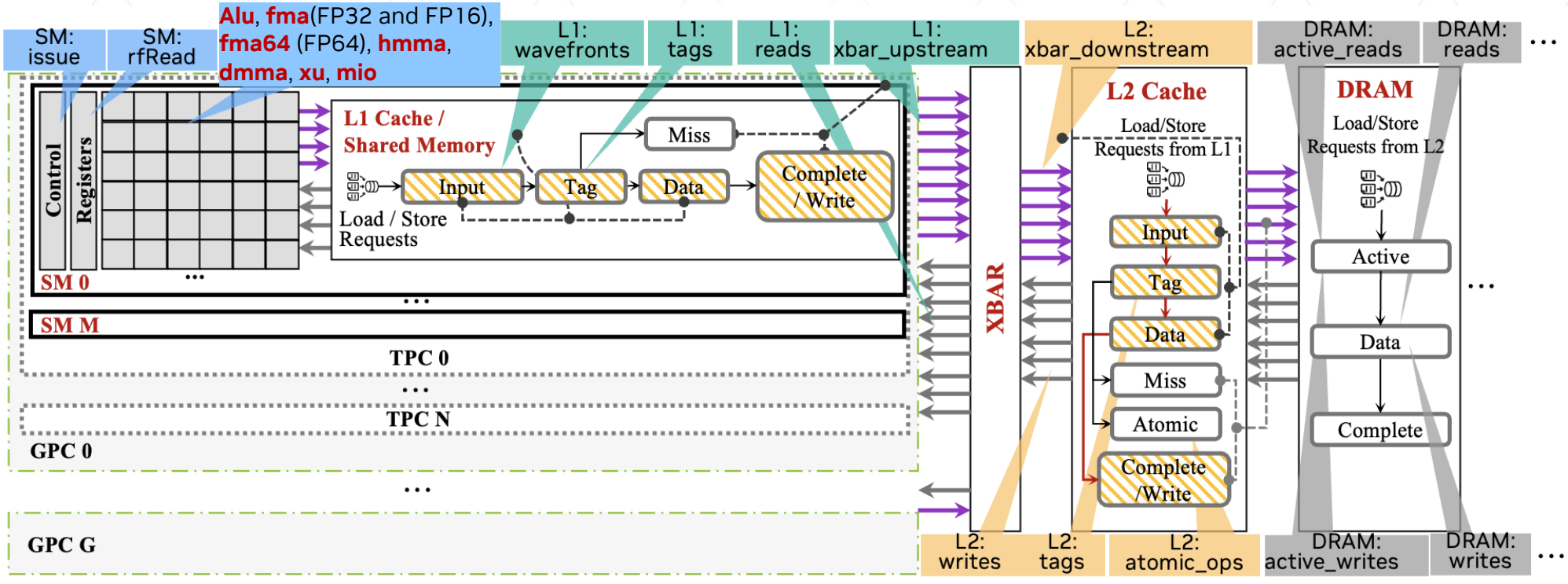


Design: Fine-grained Limiter Modeling

FTIM (Fine-Grained Traffic-Based and Imbalance-Aware Modeling)

Based on traffic from existing chips

- **Fine-Grained**: intra-unit pipeline analysis
  - Units: **SM, L1, L2, DRAM** (XBAR limiters are decomposed into L1 and L2 for analysis)

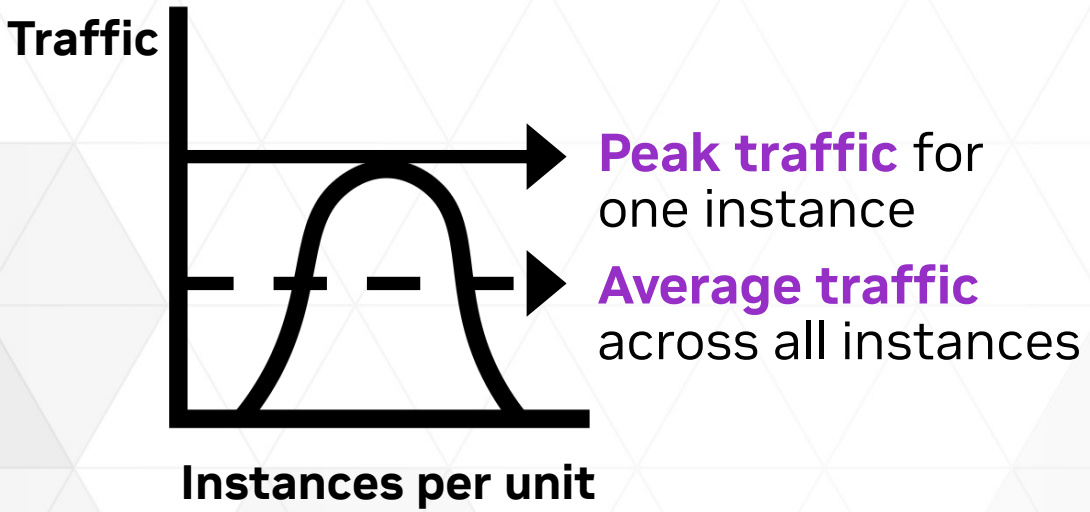


Design: Fine-grained Limiter Modeling

FTIM (Fine-Grained Traffic-Based and Imbalance-Aware Modeling)

Based on traffic from existing chips

- **Fine-Grained**: intra-unit pipeline analysis
- **Imbalance-Aware**: per-limiter cross-instance **average** and busiest-instance **peak** features



Design: Fine-grained Limiter Modeling

FTIM (Fine-Grained Traffic-Based and Imbalance-Aware Modeling)

Based on traffic from existing chips

- **Fine-Grained**: intra-unit pipeline analysis
- **Imbalance-Aware**: per-limiter cross-instance **average** and busiest-instance **peak** features
- **Frequency-sensitive**: raw cycle values **normalized to frequency**
- **Extensible** limiters and architectural units: supporting **addition** of new limiters and units

Design: SOL Ratio Prediction

How to predict SOL ratios for SM, L1, L2, and DRAM?

Characteristics

- **Intra-unit heterogeneity**: Performance constrained by **multiple distinct limiters** within each unit
- **Inter-unit dependency**: Performance coupled with **upstream and downstream** components (limiters) in the hierarchy

➡ Fit the paradigm of **multi-task learning**



Design: Limiter-Driven Runtime Generation

How to compute final execution time from all practical limiters?

Characteristics

- **Categorization-Feature Relations**: limiter importance varies drastically across **different unit-bound kernels**
- **Execution Determination**: runtime arises from **nonlinear interactions among limiters**; bounded by **hardware physics constraints**

⇨ **Workload Classification** for Unit-Bound Kernels

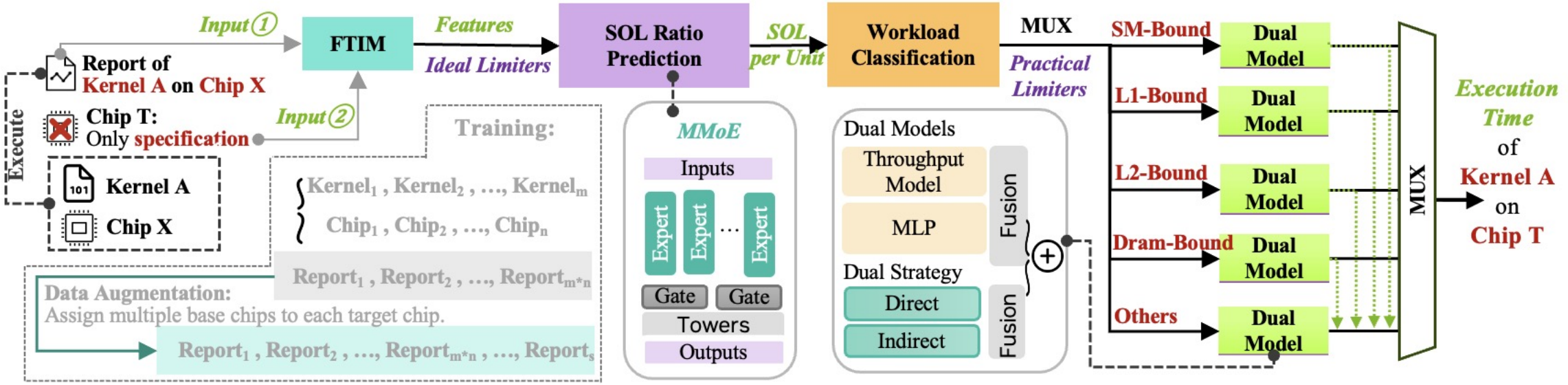
⇨ **Dual Model**

- MLP captures nonlinear correlations
- Throughput model enforces physical constraints

Design: Overview

Inputs

- **Kernel traffic** extracted from executions on the existing chip
- **Specifications** (architectural knobs) of target chips

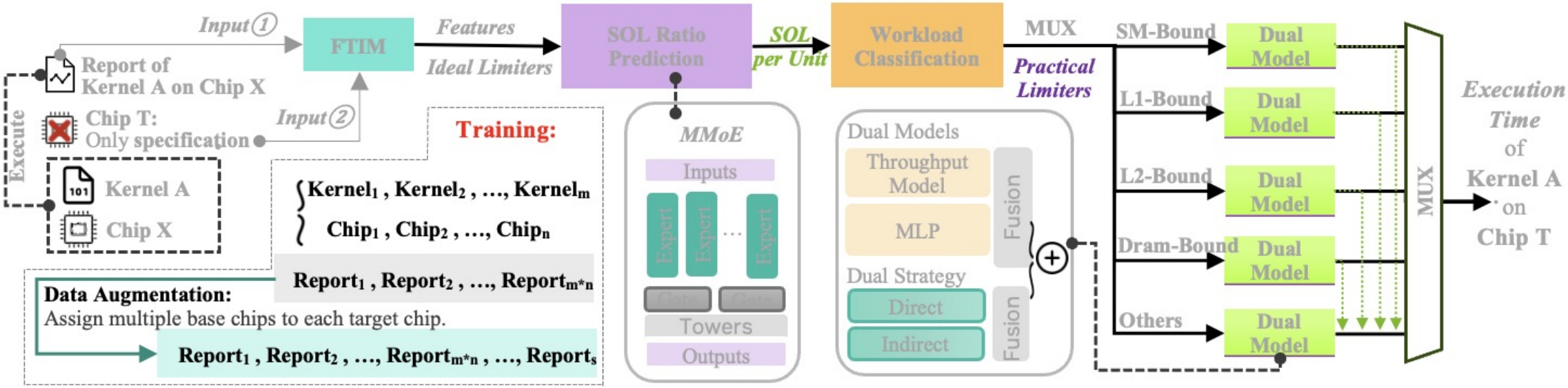


Design: Overview

Training

- For each kernel, extract traffic from executions **across diverse chips**
- For a given target chip, **train using traffic** collected from various chips

➡ Learn SOL ratio variations across kernels and chips, being traffic-shift-aware



Evaluation: Methodology

Chips:

- **Seven chips**: **GA100**, **L40**, **H100**, **GH200**, **GeForce RTX 5080**, **GeForce RTX 5090**, **B200***
  - For each chip, **GPC and DRAM frequencies** are varied
  - **MIG (Multi-Instance GPU)** generates additional partial chip configurations under different frequency settings

Kernels:

- **284 kernels** total:
  - **85% real-world kernels** from molecular dynamics, database, finance, etc.;
  - 15% synthetic microbenchmarks
- Other fake kernels to enrich the training datasets

Training/Testing Schemes:

- **B200** and GA100 are fully excluded from training and used solely for testing
- Application-level kernel split: **80% kernels for training**, the rest for testing

*B200 and GeForce RTX belong to the Blackwell family but are distinct hardware, differing in CUDA compute capability, SASS target, memory hierarchy, SM configurations, etc.

Evaluation: Metrics & Baselines

Metrics:

- **MAPE** (Mean absolute percentage error (MAPE): **Average absolute percentage difference between predicted and silicon-measured runtime**

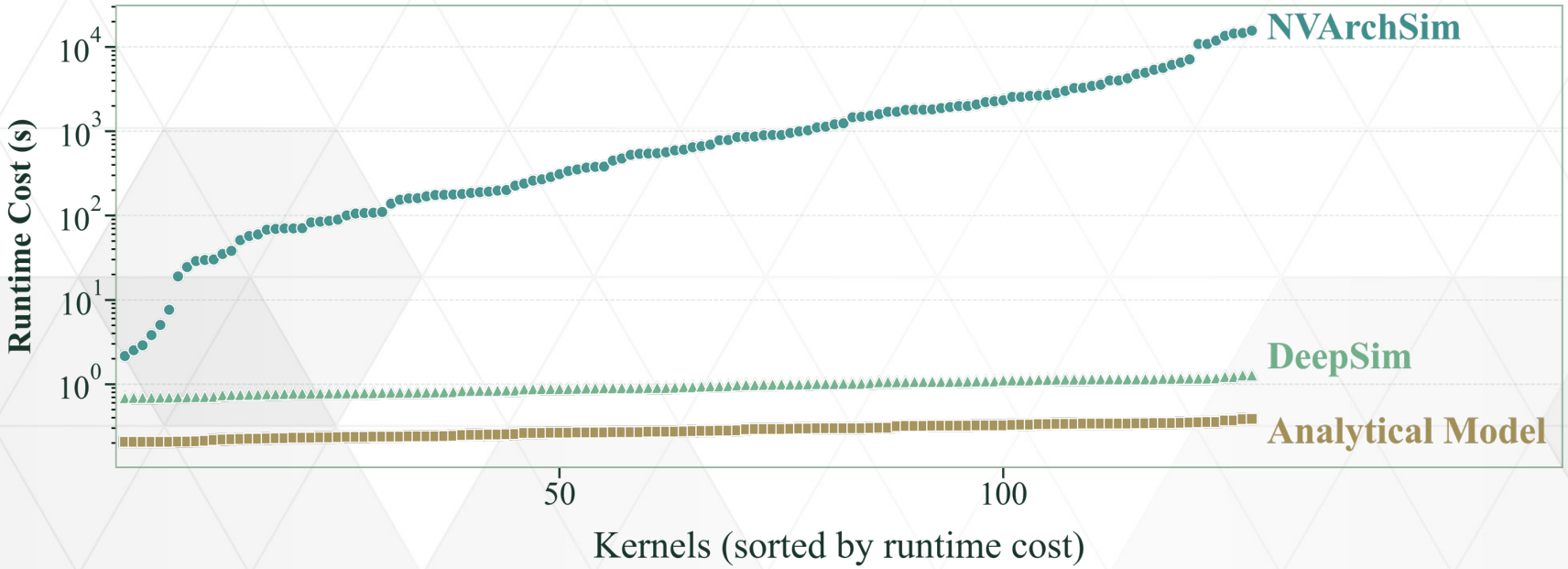
Comparison Methods:

- **NVArchSim** (HPCA'21): a representative simulation-based method
- **An analytical model** (EuroPar'24): a recent analytical performance model
- **Typical ML models**: end-to-end training and inference **using DeepSim's dataset and features**

Evaluation: Speed

Runtime Overhead on B200

- Average latency of DeepSim: **0.95 s** (single GA100)
- Time complexity: **O(1)**, matching the analytical model
- Speed comparison: **1971× speedup** of DeepSim over NVArchSim; DeepSim is 3.4× slower than the analytical model

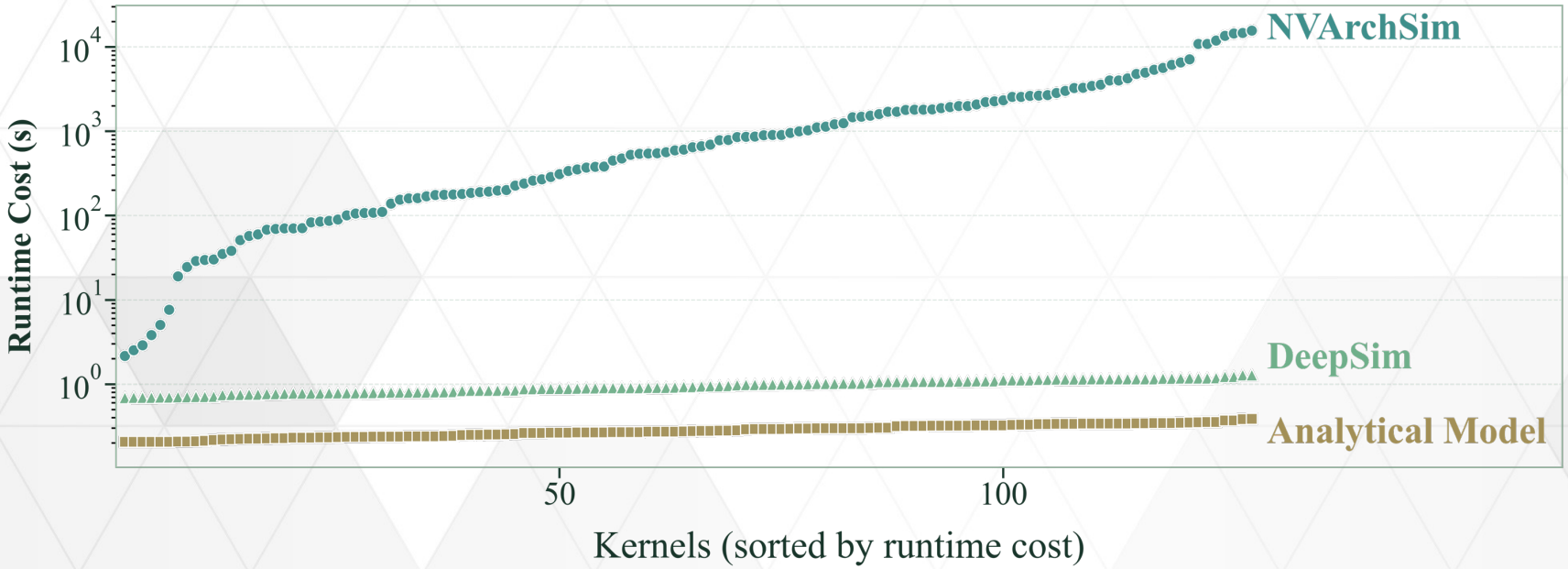


Evaluation: Speed

Fast

Runtime Overhead on B200

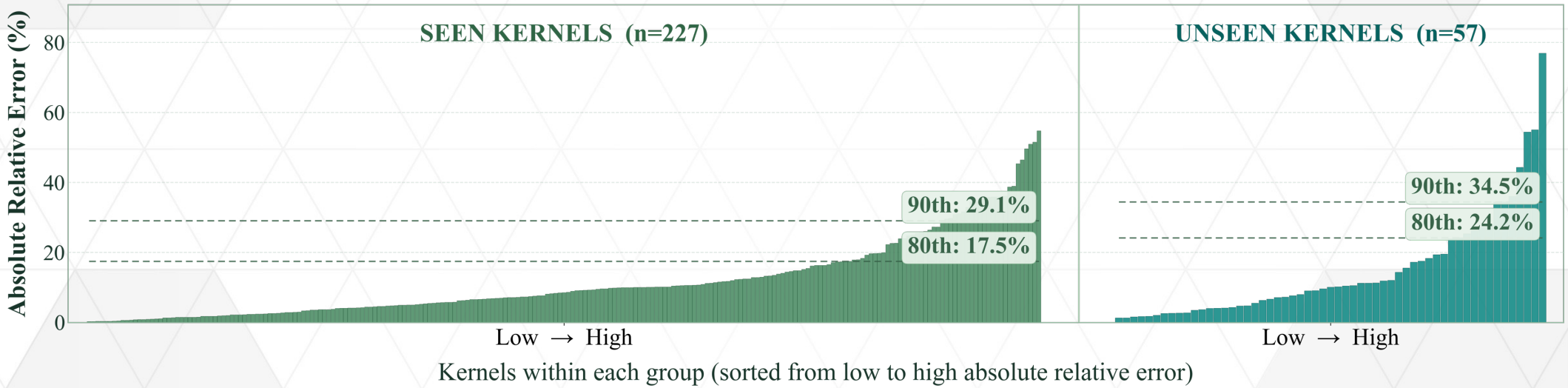
- Average latency of DeepSim: **0.95 s** (single GA100)
- Time complexity: **O(1)**, matching the analytical model
- Speed comparison: **1971× speedup** of DeepSim over NVArchSim; DeepSim is 3.4× slower than the analytical model



Evaluation: Accuracy

MAPE on B200

- Seen kernels: 11.67%; unseen kernels: 14.78%

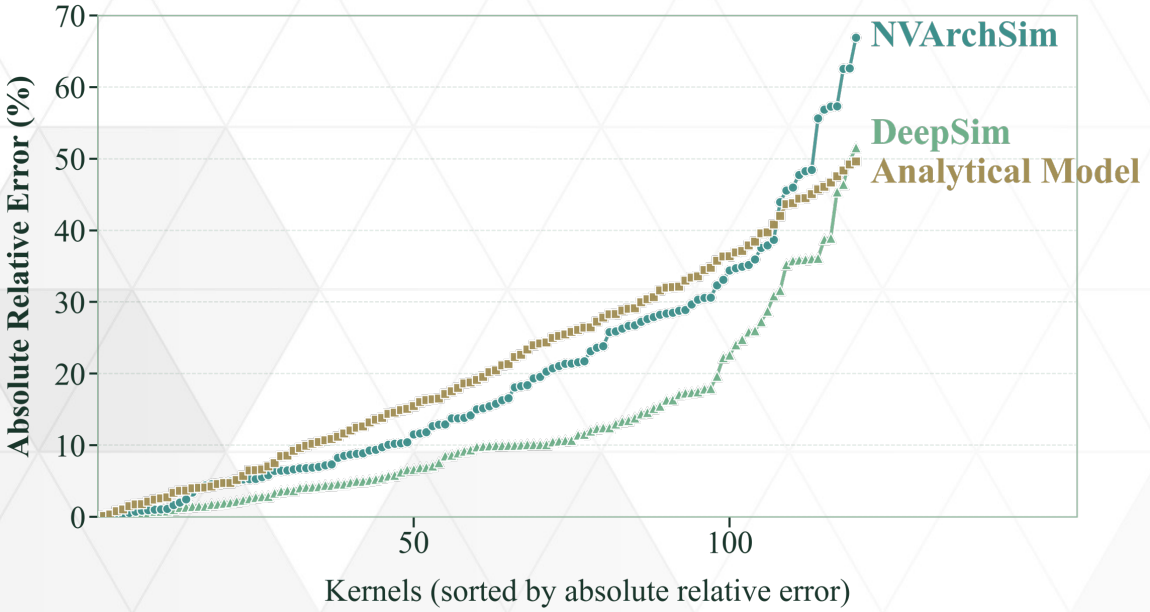


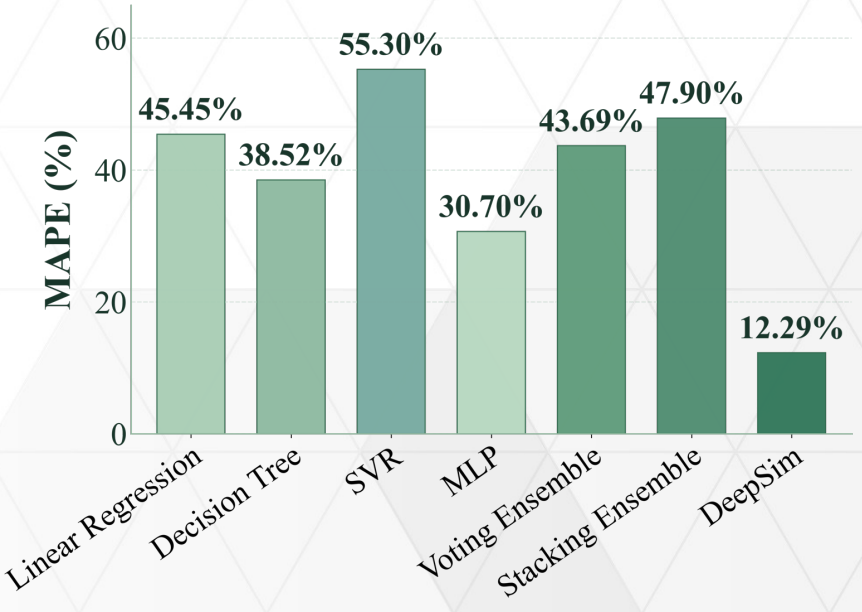
- For all kernels, 80% of kernels have error < 18.71%
- Outliers: kernels with very few CTAs, which tend to have all SOL ratios ≤ 20%

Evaluation: Accuracy

Comparison on B200

- DeepSim: **12.30%**, NVArchSim: **19.38%**, Analytical model: **21.01%** (subset of kernels runnable in NVArchSim)
- **Outperforms typical ML models** (using identical dataset & features, with hyperparameter tuning)



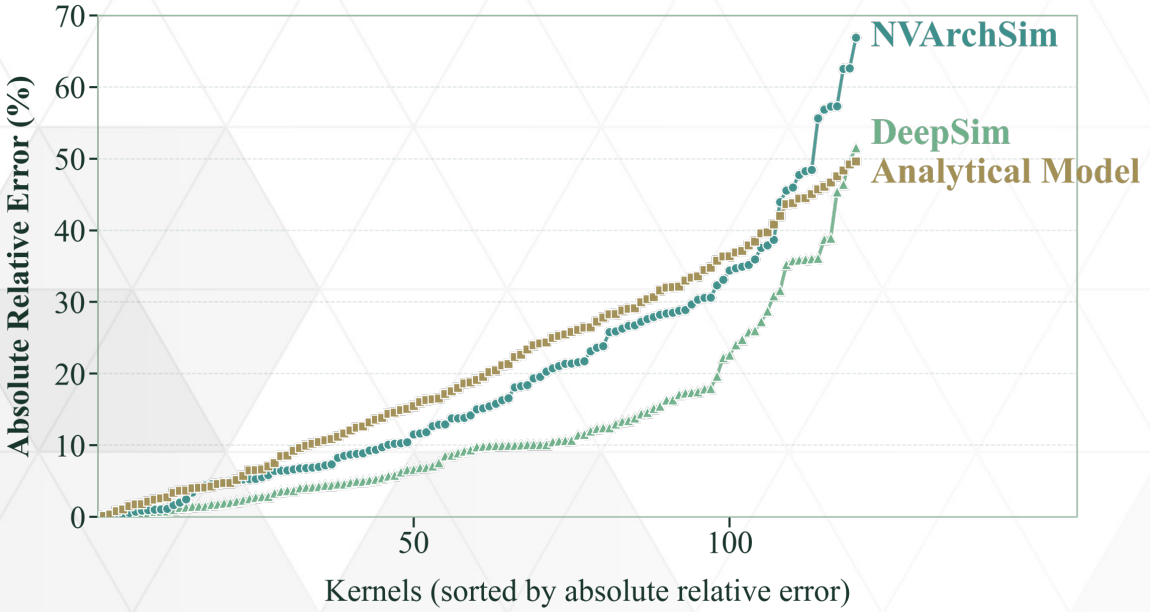


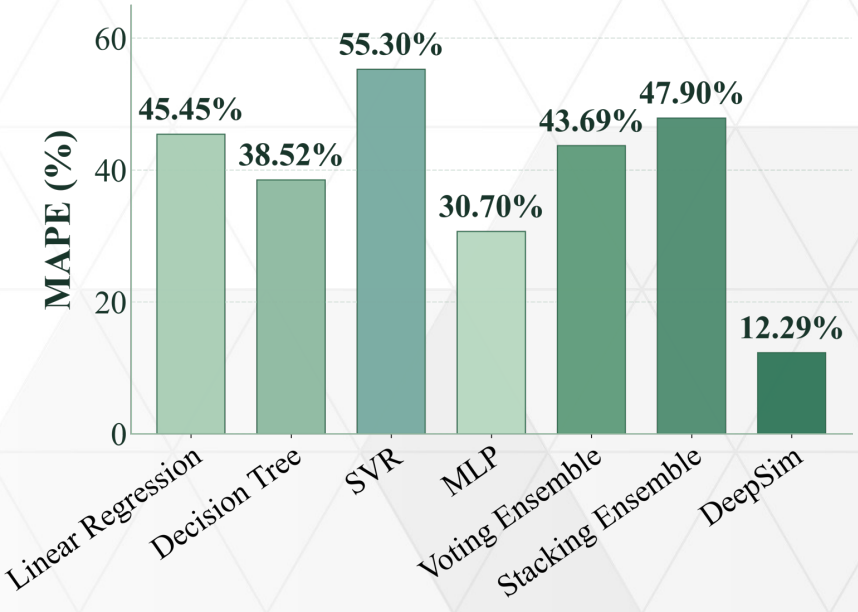
Evaluation: Accuracy

Accurate

Comparison on B200

- DeepSim: **12.30%**, NVArchSim: **19.38%**, Analytical model: **21.01%** (subset of kernels runnable in NVArchSim)
- **Outperforms typical ML models** (using identical dataset & features, with hyperparameter tuning)

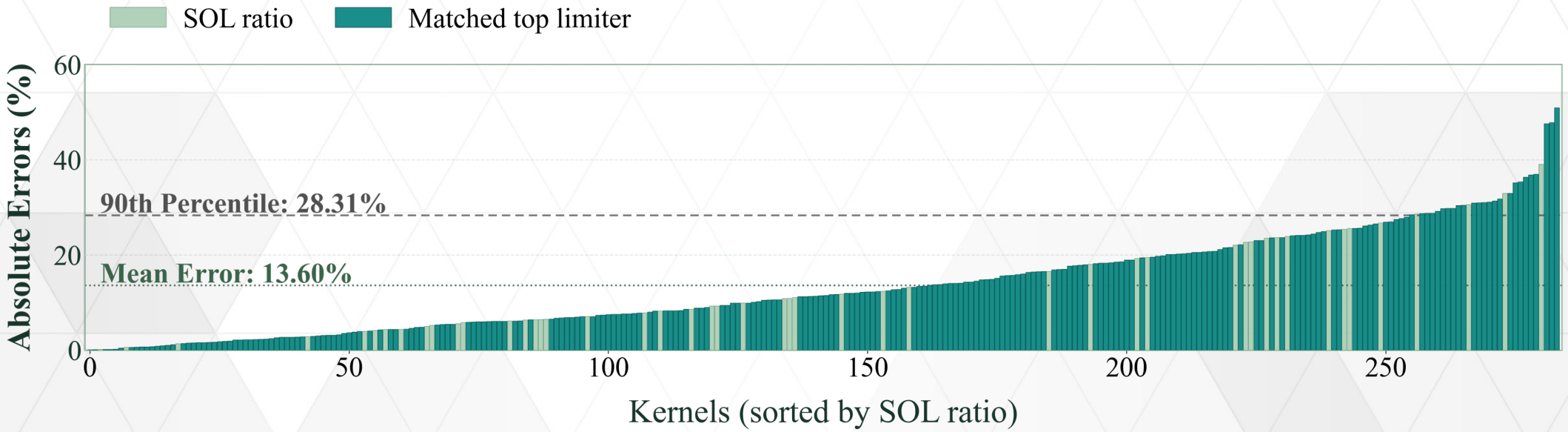




Evaluation: Explainability

SOL Ratios Accuracy and Top Limiter Matching

- **83.10% accuracy** in identifying the top limiter
- Max-unit SOL ratio prediction: **13.6% mean error**
- Per-unit SOL ratio prediction:
  - SM: 12.64%, L1: 12.06%, L2: 11.02%, DRAM: 10.91% mean error

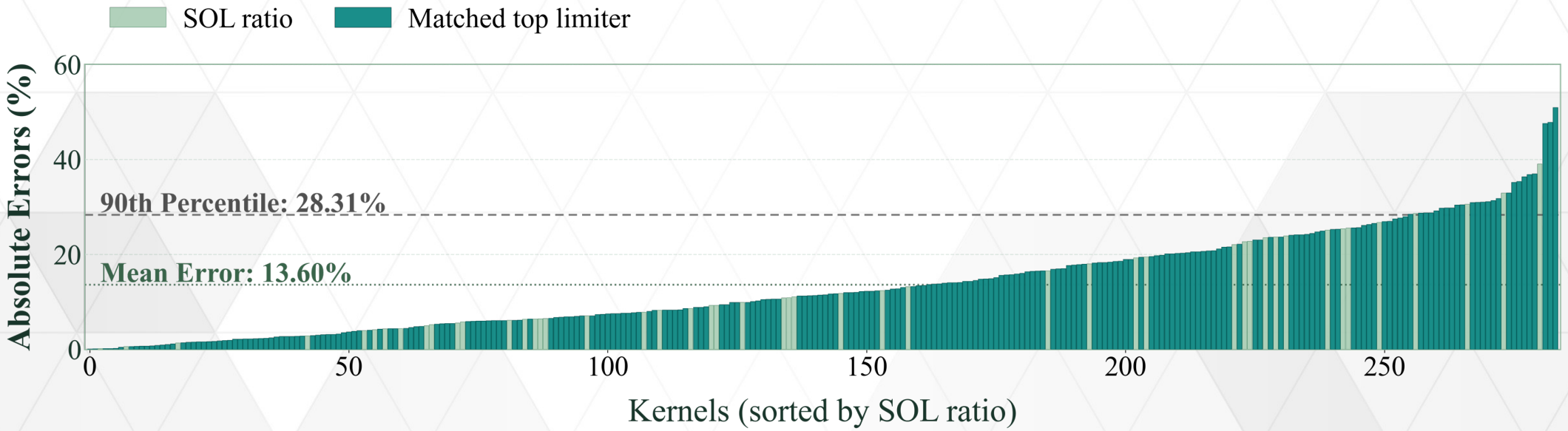


Evaluation: Explainability

Explainable

SOL Ratios Accuracy and Top Limiter Matching

- **83.10% accuracy** in identifying the top limiter
- Max-unit SOL ratio prediction: **13.6% mean error**
- Per-unit SOL ratio prediction:
  - SM: 12.64%, L1: 12.06%, L2: 11.02%, DRAM: 10.91% mean error



More in the paper

- SOL-Ratio Modeling Analysis
  - The importance of SOL-ratio for performance prediction
  - Necessity of SOL-ratio Prediction
  - Robustness of SOL-ratio to traffic shifts
- Extensibility to DL-kernels
- Component Analysis
  - Workload classification
  - Exploration of Prediction Models
- Sensitivity to base chip selection
- Validation across frequencies and MIG partitions
- Applicability across generations and vendors
- Case study on performance modeling and design space exploration
- …


DeepSim: SOL-Based Explainable GPU Performance Projection with Multi-Task Learning

Summary

Fast, **Accurate**, **Explainable** per-kernel GPU performance projection **for pre-silicon hardware**

- **Fine-Grained Traffic-Based and Imbalance-Aware Modeling (FTIM)**: analyze limiters & improve explainability
- **Multi-task learning for SOL Ratio Prediction**: boost accuracy & enhance explainability for performance projection
- **Dual Model for performance projection**: capture non-linear limiter interactions & respect physical hardware bounds

→ **DeepSim** achieves **12.19%** MAPE & **83.10%** accuracy in identifying the top limiter

DeepSim: SOL-Based Explainable GPU Performance Projection with Multi-Task Learning

Mengyue Huang[1], Zhanhui Li[1], Tim Purcell[2],
Mingqiao Li[1], Zhengru Wang[1], Nirmal Saxena[2]

[1]NVIDIA, Shanghai, China
[2]NVIDIA, Santa Clara, California, USA

