AMALI: An Analytical Model for Accurately Modeling LLM Inference on Modern GPUs

Shiheng Cao[1], Junmin Wu[1], Junshi Chen[1], Hong An[1], and Zhibin Yu[2].
[1]USTC [2]SIAT

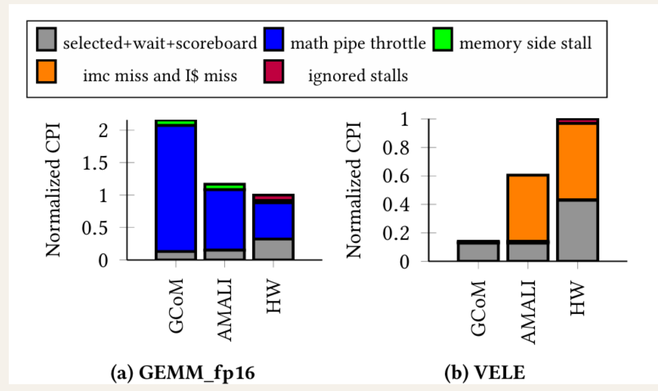
ISCA 2025

Motivation

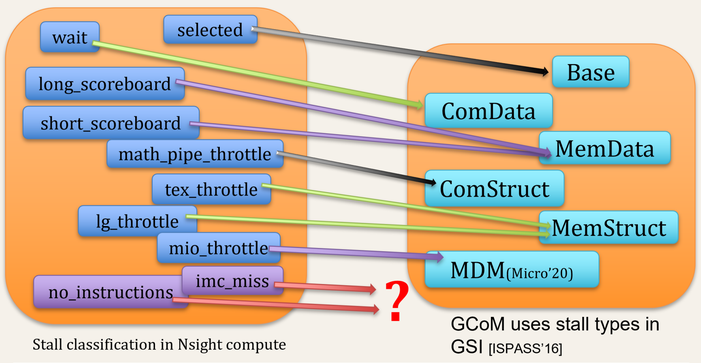
- **LLM Inference is GPU-Intensive**
- **Fast & Insightful GPU Modeling is Needed**

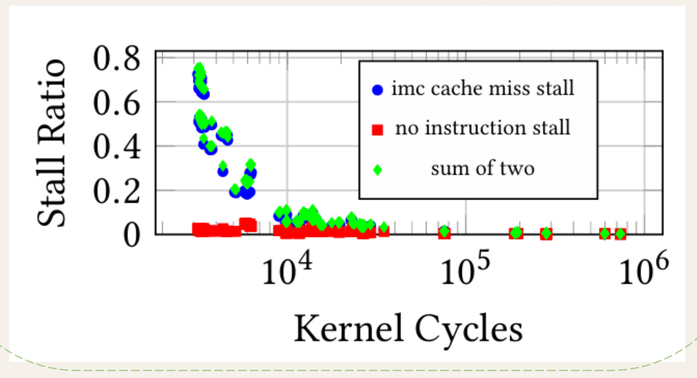
Simulator	DL-based Model	Analytical Model
Slow	Needs data	Fast
Complex	Black-box	Interpretable
Accurate	Architecture dependent	Insightful Architecture-aware

- As LLM inference rapidly evolves, tools that quickly and insightfully identify GPU bottlenecks are essential.
- Existing GPU analytical models fall short of modeling LLM inference performance.

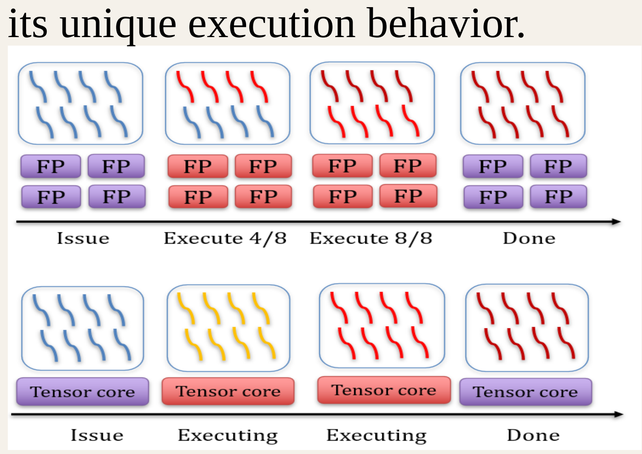


- **IMC cache miss** and **instruction cache miss** are ignored. They account for large stall ratio in **LLM inference.**

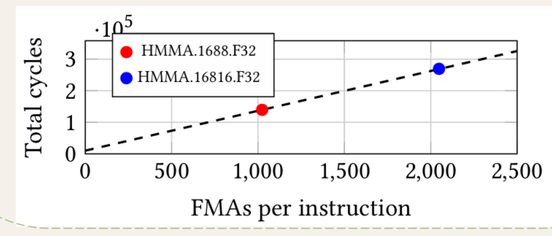




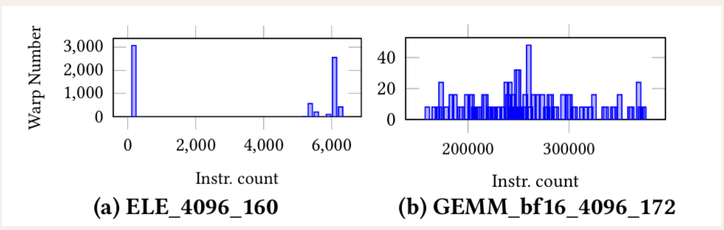
- Accurate **Tensor Core model** is essential for modeling **LLM inference**.
- However, prior methods fail to capture its unique execution behavior.

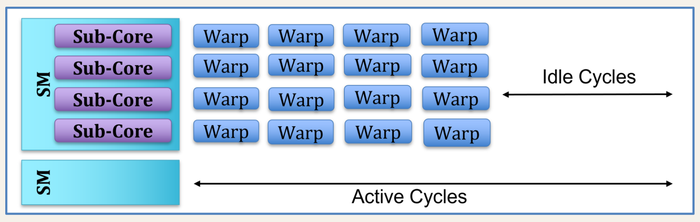


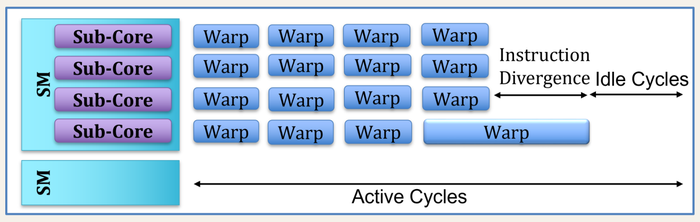
- Instruction modifier influence latency.



- Assuming **uniform warp behavior**, as in prior models, leads to inaccurate modeling of LLM inference workloads.



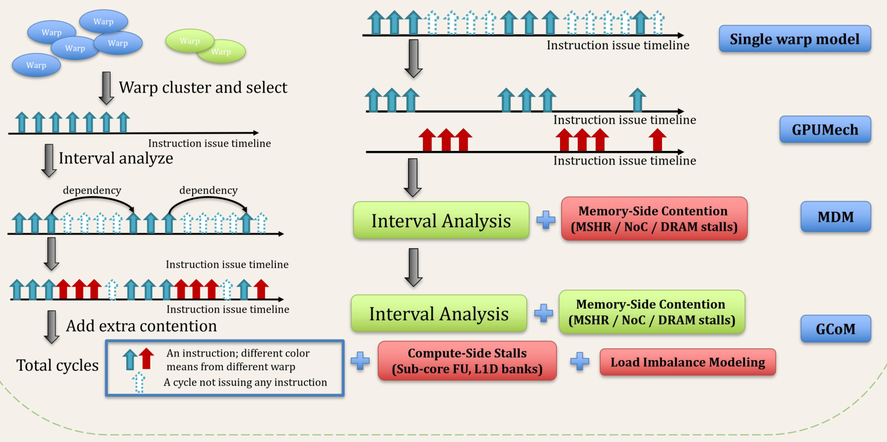




AMALI Overview

Interval-based GPU Analytical Models

- Use a **single representative warp** to estimate kernel behavior.
- Split execution into **intervals** with stall analysis
  - AMAT from cache simulation.
  - Instruction latency from microbenchmark.
- Based on this, prior works add more contention.



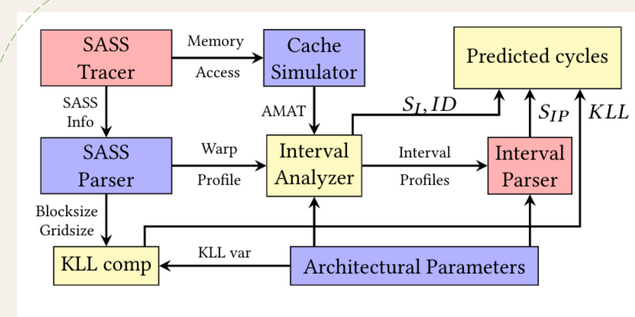


Figure 6: An overview of AMALI. SASS - CUDA assembly instruction. AMAT - Average Memory Access Time. KLL - Kernel Launch Latency. S_I - Results of interval analyzer. ID - Instruction Divergence.

Model **tensor core** with throughput and modifier

$$\text{initiation\_interval(II)} = \frac{\text{FMA\_count}}{TP_{dt}}$$

$$Latency_{TC} = II_{TC}$$

Model **launch latency** with equation corresponding to gird size and block size

$$KLL = s \cdot GS + \mathbf{k}$$

$$s = \alpha \cdot (BS)^2 + \beta \cdot (BS) + \gamma$$

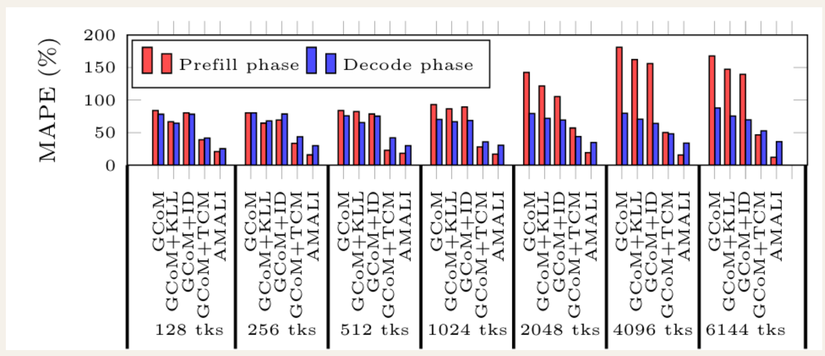
Model warp **instruction divergence** by difference between representative warp and largest one

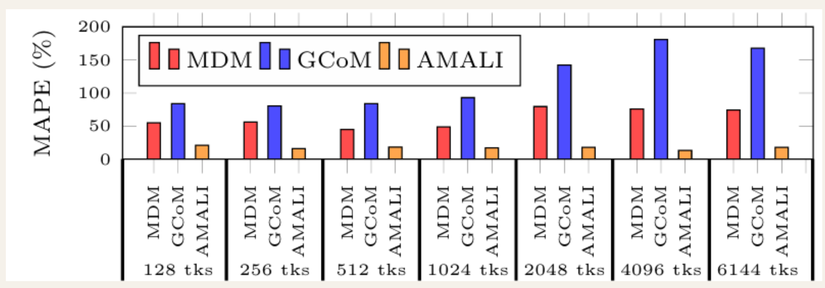
$$ID = (\text{max\_sub-core\_Instr} - I_{SC\_Repr.warp})/IssueRate$$

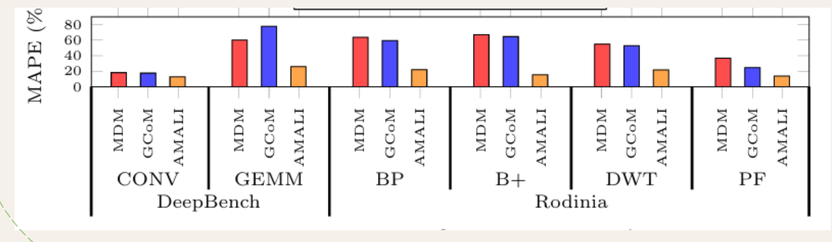
- **SASS Tracer**
  Dynamically collects SASS instruction traces from real GPU execution with NVBit.
- **SASS Parser**
  Decodes SASS instructions to extract operation type and data dependencies.
- **Cache Simulator**
  Models cache behavior to get average memory access time.
- **Interval Analyzer**
  Divides warp execution into intervals based on data dependencies and issue constraints.
- **Interval Parser**
  Analyzes each interval to catch stalls caused by compute/memory throughput.

Evaluation and Design Space Exploration

Modeling LLM Inference and other benchmarks







- Modeling Llama3 inference shows that AMALI achieves MAPE reduction from 127.56% to 23.59% against GCoM, and ablation confirms the efficacy of each individual improvement.
- Varying prompt lengths confirm AMALI's scalability vs. GCoM and MDM.
- DeepBench & Rodinia results show AMALI works across AI and general workloads

Design Space Exploration

- Kernel cycles with break-downs predicted by AMALI of on A100 and H100, and the measured cycles. X axis represents the five GEMM ($M \times K \times N$) kernels. Each block partitioned by the red dash lines contains two bars, and the left and right ones denote the cycles consumed by the same kernel running on A100 and H100, respectively.
- Modeled A100 = A100 SASS traces + **A100 config**
- Modeled H100 = A100 SASS traces + **H100 config**