14th Annual MVAPICH User Group (MUG) Conference

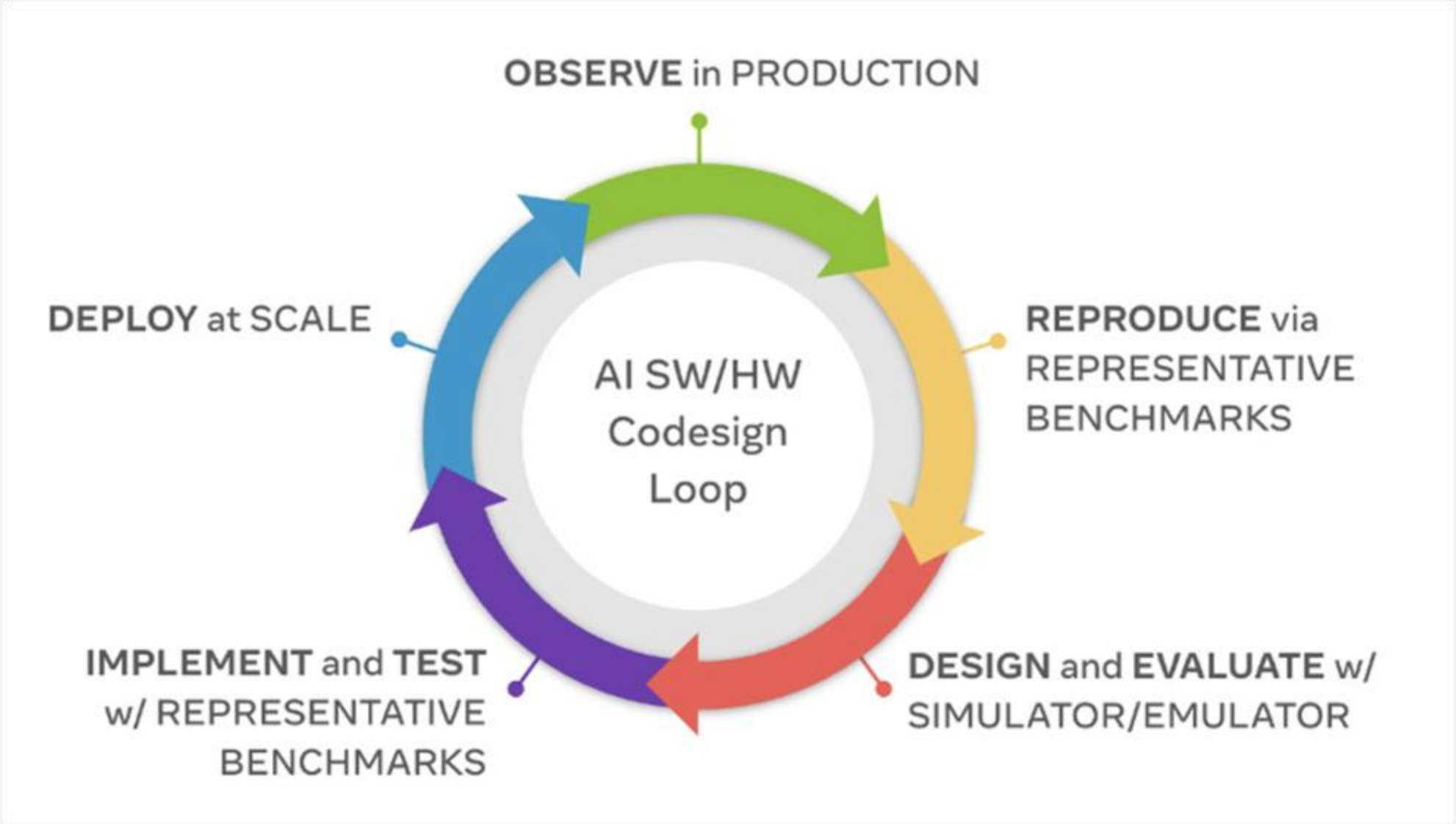
Predictive Performance Modeling and Synthetic Workload Generation for Parallel DNN and LLM Training

Mihai Gabriel Constantin
University Politehnica of Bucharest

Dan Mihailescu
Keysight Technologies

The HW/SW Co-design Cycle

Representative workload behavior connects software development with hardware design

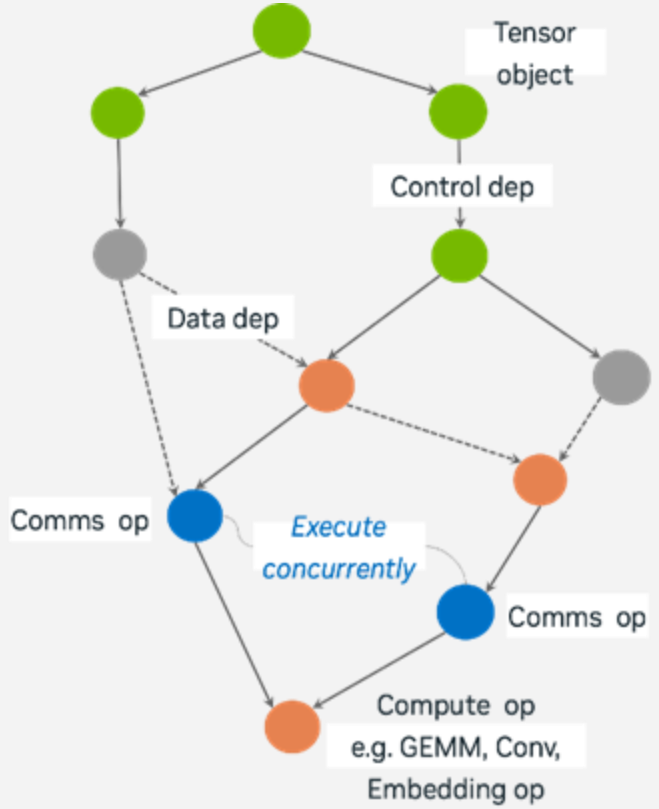


MLCommons Chakra

A portable, graph-based workload representation

Goal: “A common graph format for co-design”

- **Chakra Execution Trace (ET)**: a portable description of how a distributed AI workload behaves - compute, memory, communication, and dependencies.
- Captures what a workload **does** — without requiring access to model weights, datasets or proprietary source code.
- Co-developed by Meta and Georgia Tech in 2023; standardized under MLCommons as an open research working group.
- Today: 40+ member working group — hyperscalers, compute vendors, research groups, simulation and testing vendors.



*https://engineering.fb.com/2023/09/07/networking-traffic/chakra-execution-traces-benchmarking-network-performance-optimization/

Three Research Questions

Collaboration between Keysight and University Politehnica of Bucharest

1. Can operation durations be predicted **across world sizes**?

    Scaling Prediction Model

2. Can valid Chakra traces be generated **without executing the workload**?

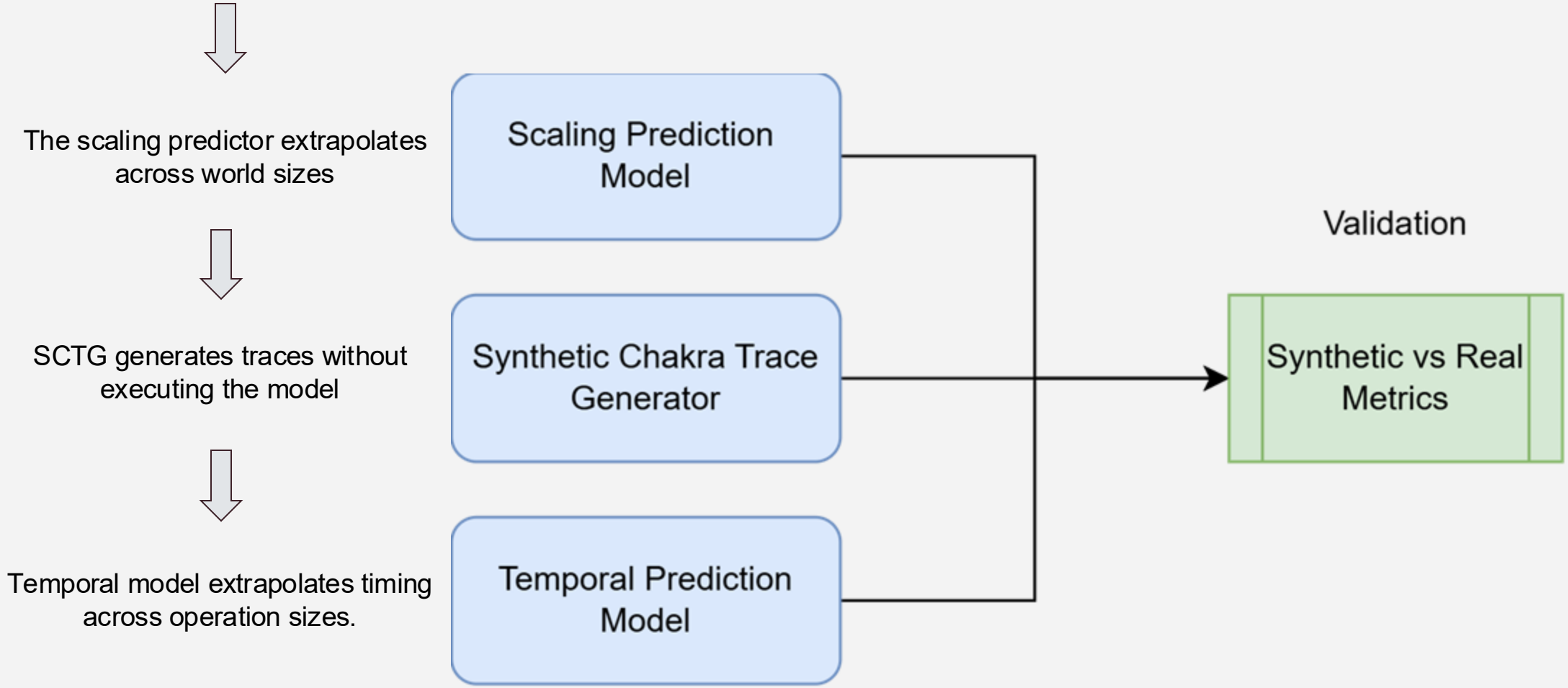
    Synthetic Chakra Trace Generator

3. Can **timing** be extrapolated **to unseen operation sizes on the same GPU**?

    Temporal Prediction Model

Answering the research questions

Chakra produces real traces



SCALING PREDICTION MODEL

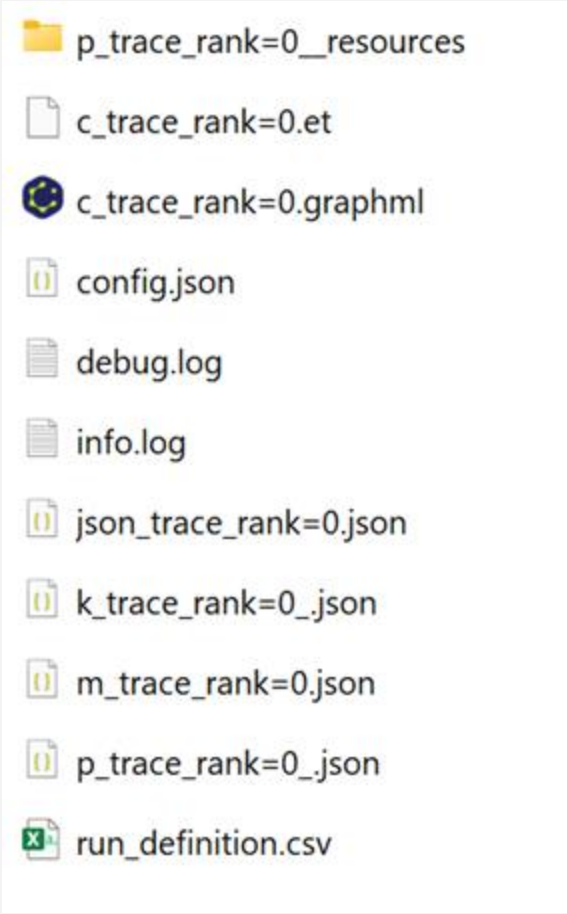
Mihai Gabriel Constantin, Dan Mihailescu - University Politehnica Bucharest, Keysight Technologies

Scaling Prediction Model

Experimental example

Experiment example:

- Scale a training process that runs on 2 nodes



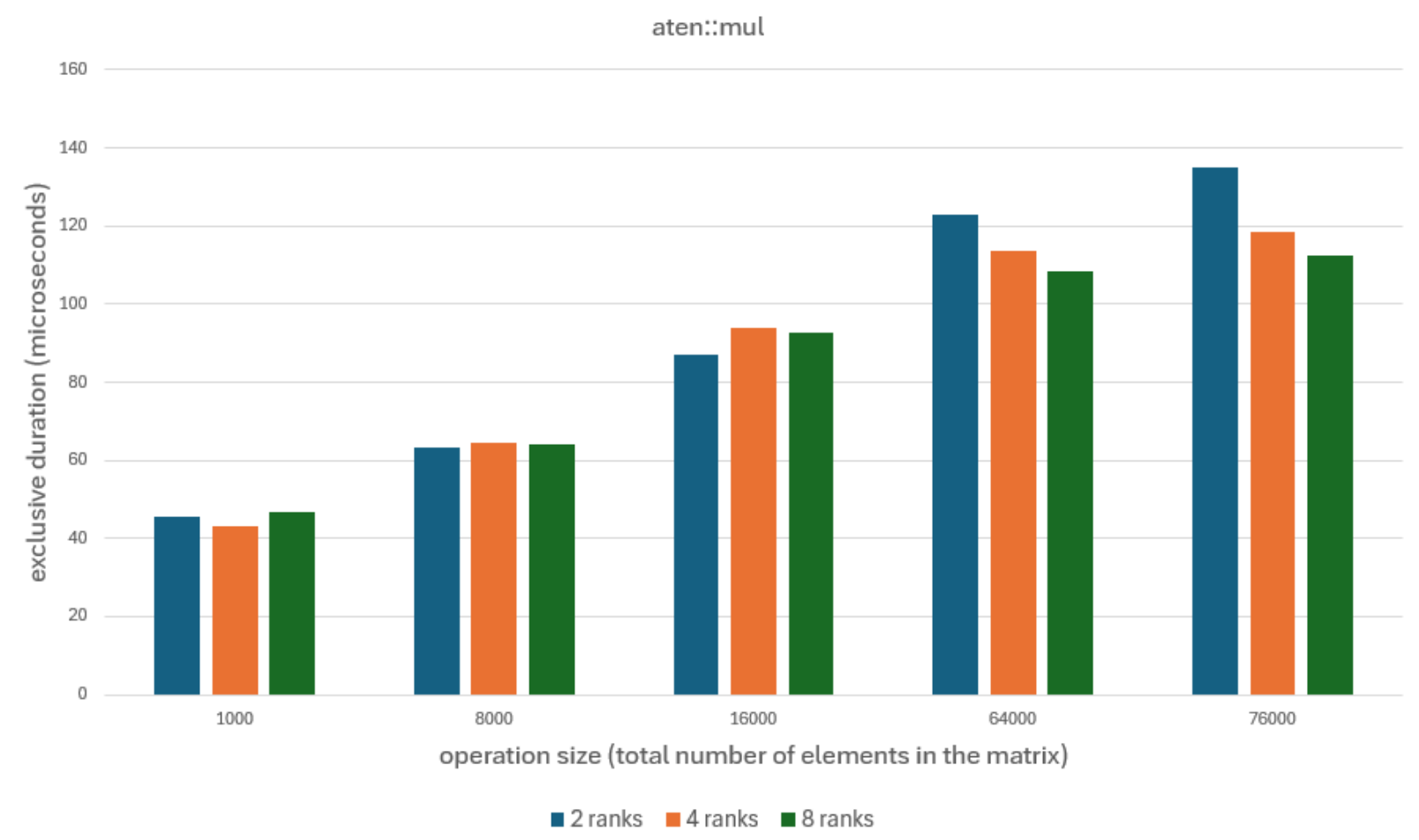
- To a training process that would run on

4 / 6 / 8 /16 nodes

Linear correlation analysis

Operation latency does not scale linearly with world size

Experiment: **is there a simple linear correlation between duration and world size?**



Scaling Prediction Model

Metrics for scaling prediction accuracy

Computing the performance:

- *E* metric, computed as the mean absolute percentage error across evaluated trace nodes.
- *E_max* metric, maximum node-level E value.

$$E = \frac{1}{N} \sum_{i} \frac{|t_{\text{pred},i} - t_{\text{real},i}|}{t_{\text{real},i}}$$

$$E_{max} = max(\frac{|t_{\text{pred},i} - t_{\text{real},i}|}{t_{\text{real},i}})$$

- Each experiment was repeated ten times; the reported values are aggregated across those runs.
- Evaluation covers more than 50 DNNs / LLMs and their variations.

Scaling Prediction Model

Experimental results

- Hardware setup: 2, 4, 6, 8, 16 nodes, 2080 Ti, A5000, AWS T4

Experiment	No. of nodes	Hardware	Data	E	E_max
Ex1	2, 4	2080 Ti	Initial dataset	11.37%	31.10%
Ex2	2, 4	2080 Ti + A5000	train on Ex1 + Ex2	10.95%	29.08%
Ex3	2, 4, 6, 8	2080 Ti + A5000 + T4	train on Ex1 - Ex3	9.21%	26.24%
Ex4	2, 4, 6, 8, 16	2080 Ti + A5000 + T4	train on Ex1 - Ex3, test on 16 nodes	10.23%	27.30%
Ex5	2, 4, 6, 8, 16	2080 Ti + A5000 + T4	train on Ex1 - Ex5	9.31%	25.91%
Ex6	**2, 4, 6, 8, 16**	**2080 Ti + A5000 + T4**	**additional data and DNNs**	**10.82%**	**28.31%**

- Mean prediction error is 9.2–11.4%; worst-case node error remains 25.9–31.1%.

SYNTHETIC CHAKRA TRACE GENERATOR (SCTG)

Mihai Gabriel Constantin, Dan Mihailescu - University Politehnica Bucharest, Keysight Technologies

Synthetic Chakra Trace Generator

Long-term goals:

- Generate realistic execution traces for PyTorch AI/ML model architectures
- Simulate time and performance based on GPU type and operation size
- Vary training parallelism mechanism, input size and the number of model params

Currently validated:

- Validated for Distributed Data Parallel and Fully Sharded Data Parallel
- Timing models are currently validated for A5000, 2080 Ti, T4 GPUs.

Not yet supported:

- External/custom code, other parallelism strategies, or untested primitives

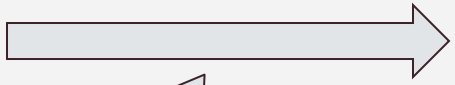
Synthetic Chakra Trace Generator

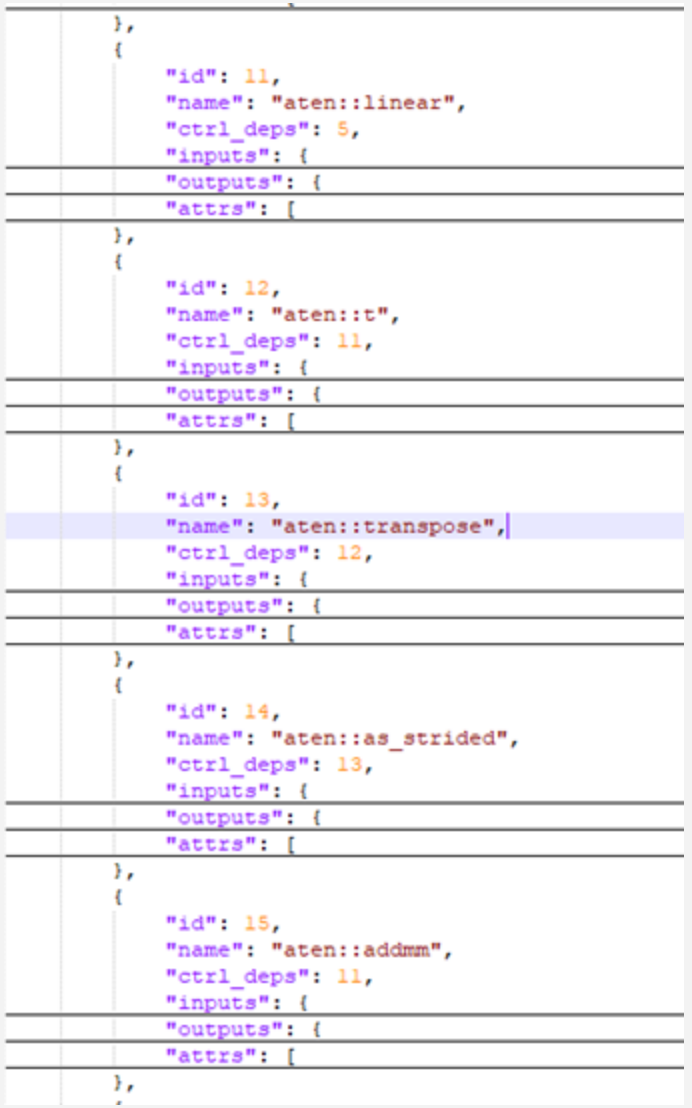
General principles of synthetic trace generation

This layer does an affine linear transformation for the input:

$$y = xA^T + b$$

```
x = x.view(-1, 784)
x = self.fc1(x)
return F.log_softmax(x, -1)
```

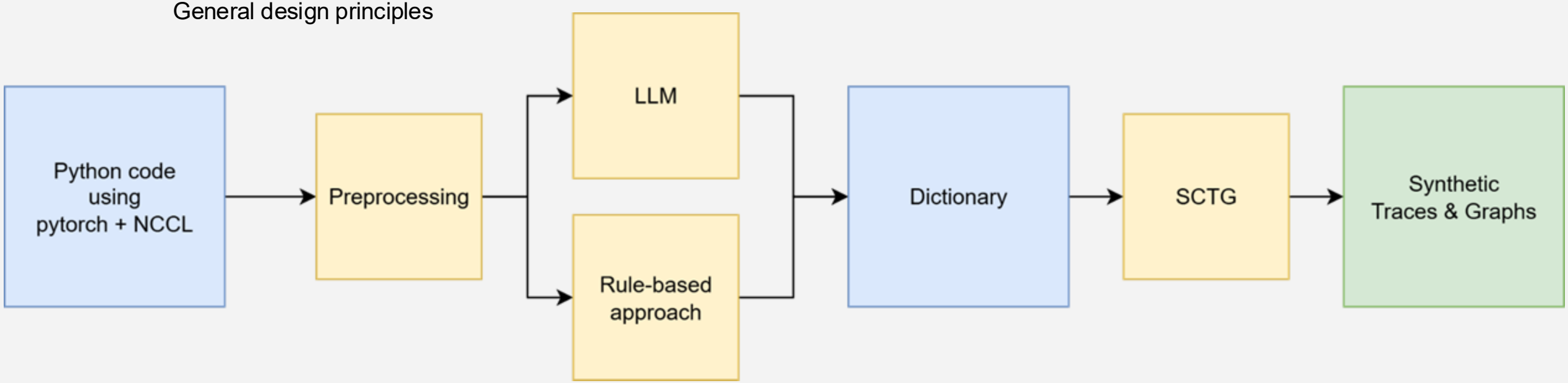




1. Analyze each layer without executing the complete DNN on physical GPUs.
2. Generate operations for multiple ranks.
3. Produce Chakra-format execution traces directly from the model code.

Synthetic Chakra Trace Generator

General design principles

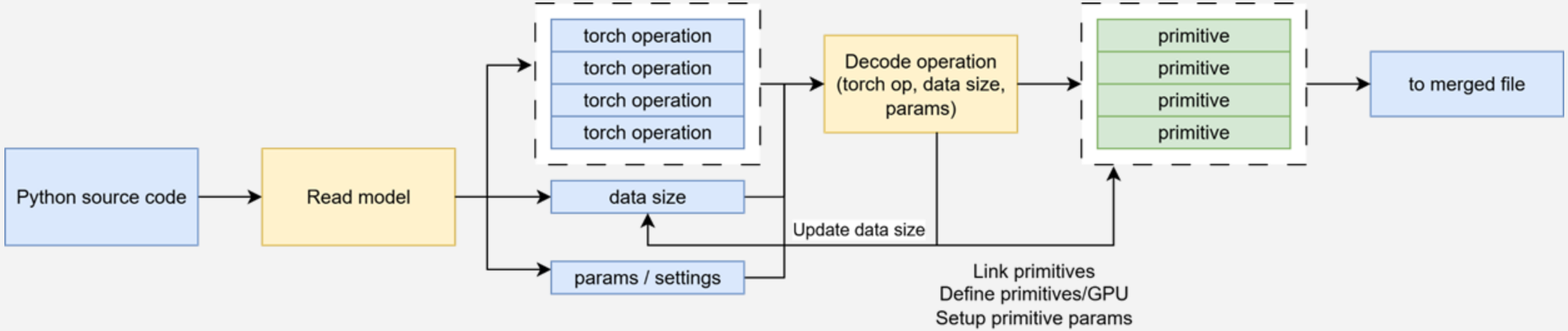


Method

- The current implementation uses **rule-based preprocessing**.
- LLM-assisted source-code analysis may be added for workloads that cannot be handled by the rule-based parser.

Synthetic Chakra Trace Generator

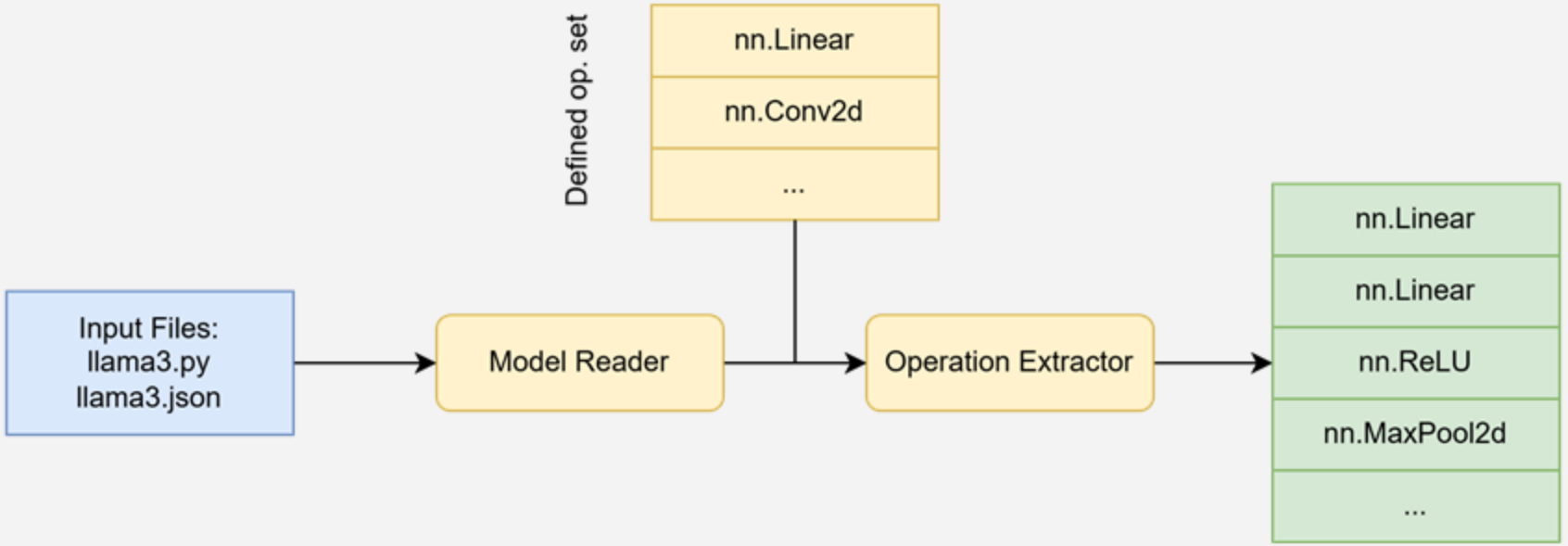
SCTG high-level architecture



Synthetic Chakra Trace Generator

SCTG high-level architecture

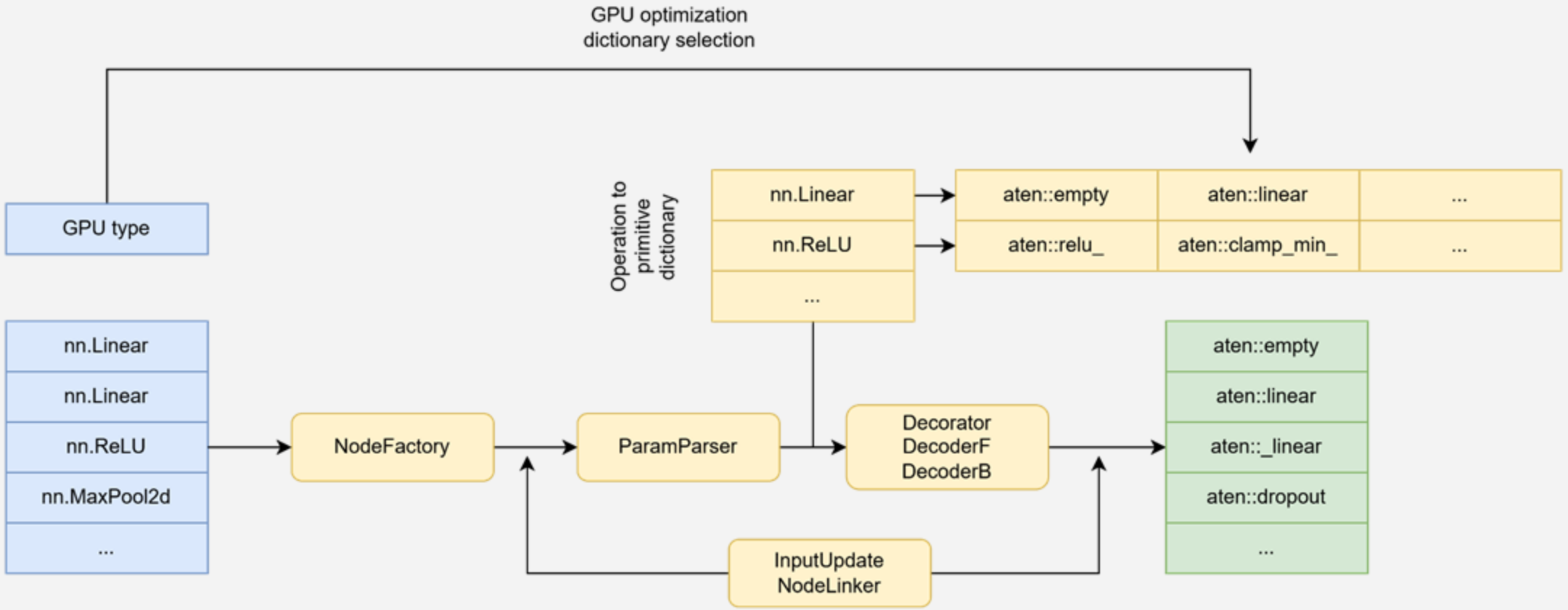
Model Reader



Synthetic Chakra Trace Generator

SCTG high-level architecture

Node Builder



Synthetic Chakra Trace Generator

Metrics and experimental results

Metrics

- Graph similarity - Zhang-Shasha Tree Edit Distance (*ZSS*):
  - It can process tree-like / graph structures from a topological standpoint
  - Checks the number and identity of nodes
  - Checks the connection between nodes
  - Checks node names (the primitives that are being called)

Same testing setup (GPUs, AI models) as the Scaling Prediction Model

Results:

- Across 180 evaluated trace pairs, the generated graphs achieved a **ZSS distance of zero** for topology and node labels.

TEMPORAL PREDICTION MODEL

Temporal Prediction Model

Defining the research problem

Research Problem: The current implementation does not yet accurately predict timing for previously **unseen operation sizes**, even when measurements are available for the same GPU architecture.

Create a framework that will allow us to measure and annotate GPU performance, which will allow us to **extend the experiments to additional GPU architectures**.

Temporal Prediction Model

Methodology

Methodology:

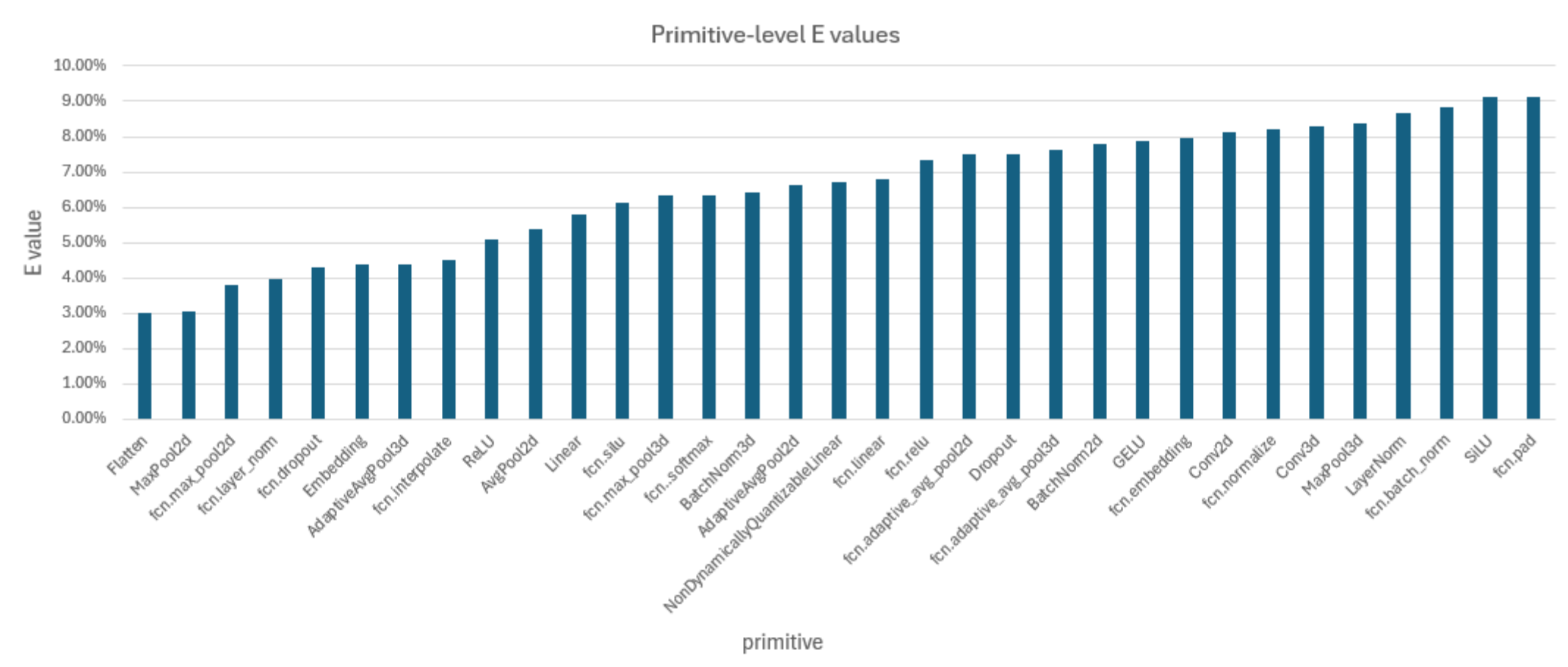
- using the same hardware setup
- run 10 times using the same GPU, average times over those runs
- do the same for a series of DNNs and LLMs
- compare the execution times of the primitives **- 32 primitives in total**

Goal:

- have an E-metric error of under 10%
- indicating low enough variability in order to correctly predict timing information
- perform an analysis of each primitive

Temporal Prediction Model

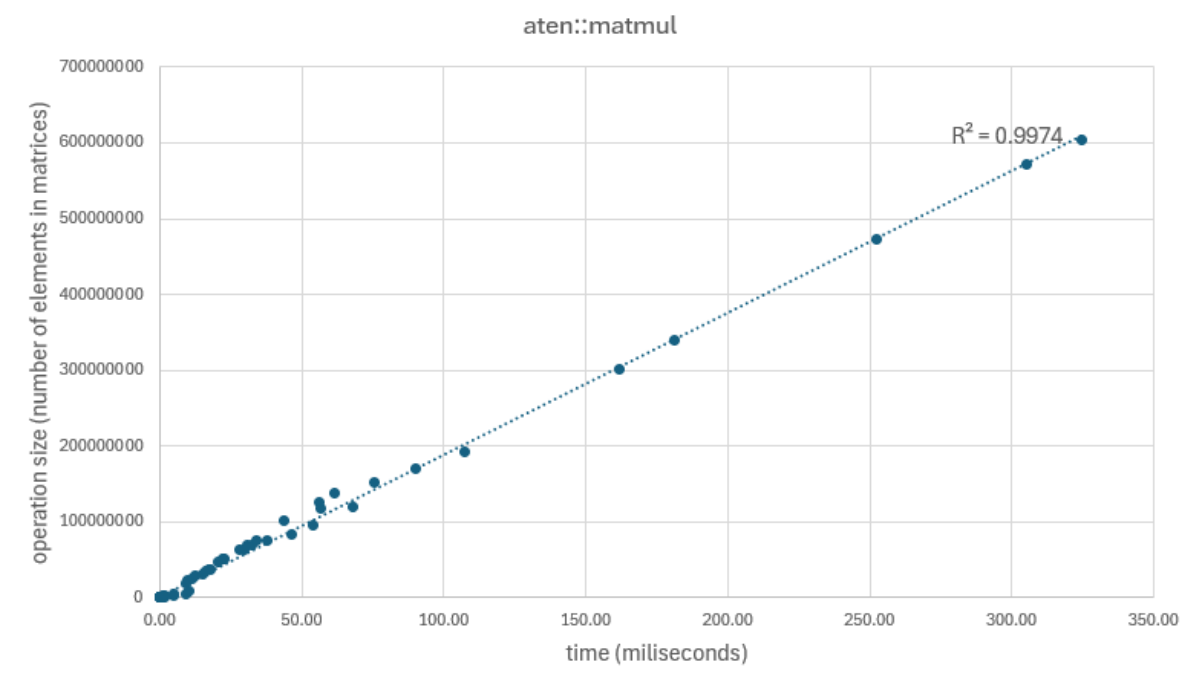
Experimental results



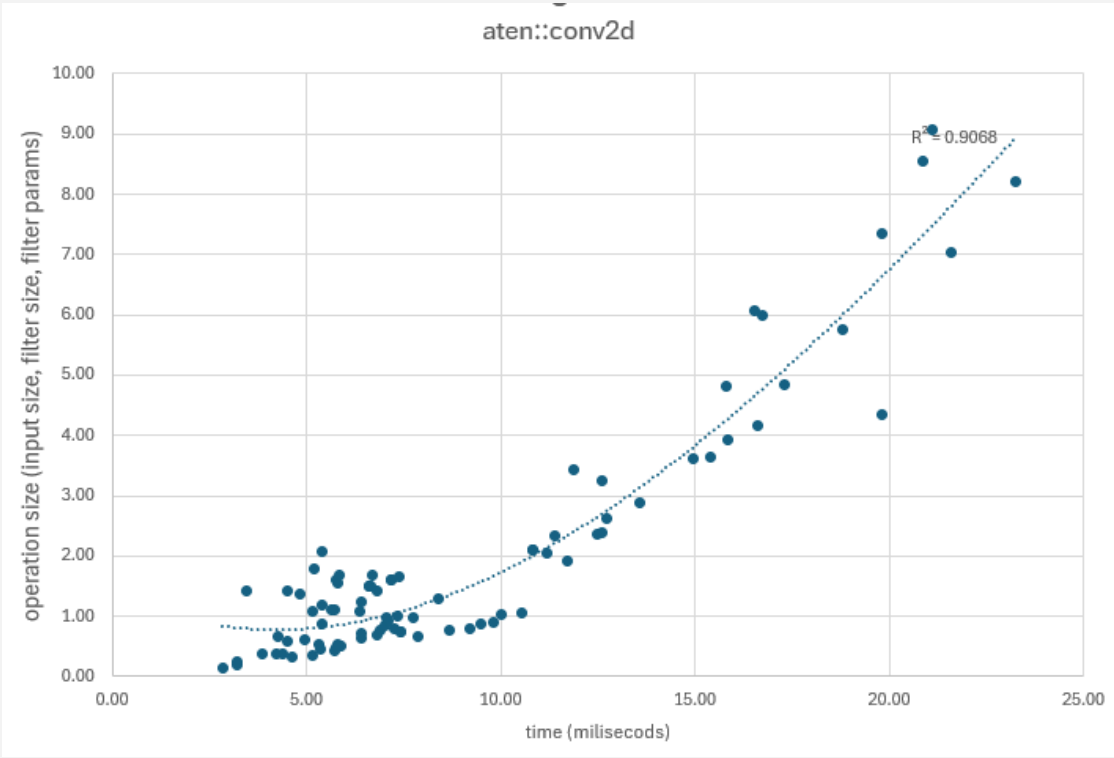
We obtained errors **between 3.01% and 9.14%**

Temporal Prediction Model

Two examples of how primitives scale up with operation size



Linear correlation between size and time



Polynomial correlation between size and time

Mihai Gabriel Constantin, Dan Mihailescu - University Politehnica Bucharest, Keysight Technologies

Conclusions

What we demonstrated:

- Scaling prediction: **9.2–11.4% mean error** across the reported experiments.
- Synthetic generation: **zero topology distance** across the evaluated trace pairs.
- Temporal calibration: primitive-level errors **between 3.01% and 9.14%** for the reported operation-size ranges.

THANK YOU!