LLMServingSim: A HW/SW Co-Simulation Infrastructure for LLM Inference Serving at Scale

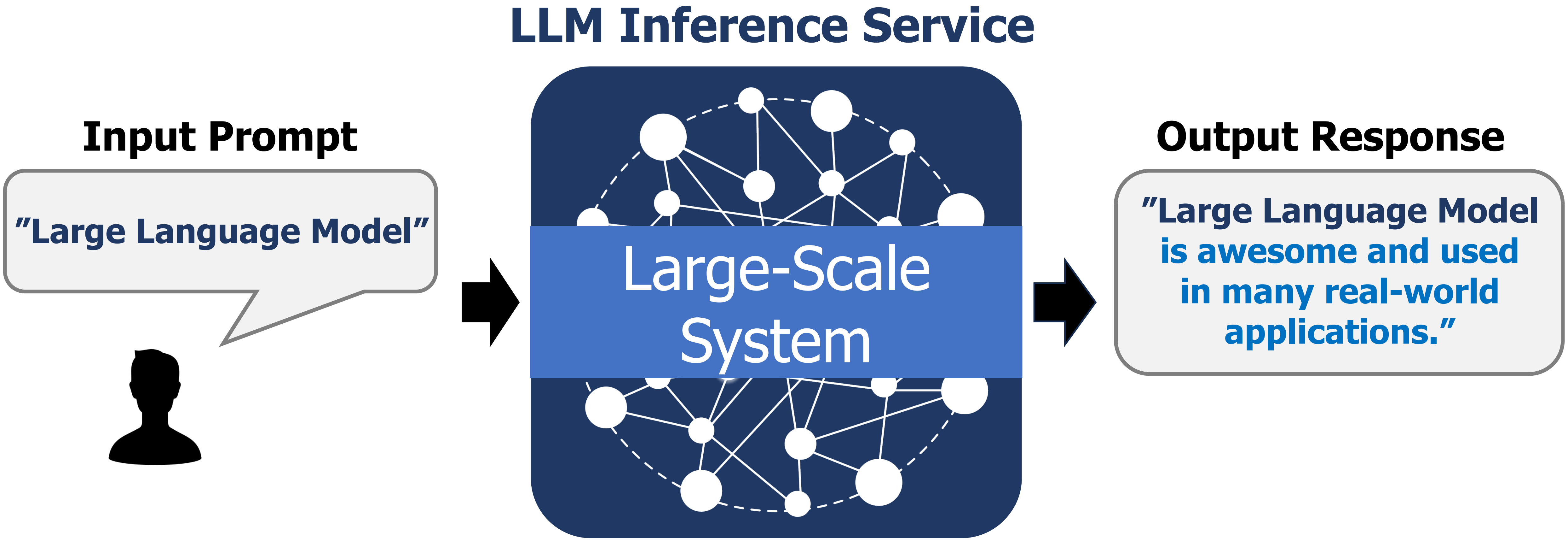
Jaehong Cho
Minsu Kim
Hyunmin Choi
Guseul Heo
Jongse Park

KAIST

KAIST

IISWC 2024

LLM Inference Serving



LLM Inference Service

Input Prompt

”Large Language Model”

Large-Scale System

Output Response

”Large Language Model is awesome and used in many real-world applications.”

Systems for LLM Inference Serving

GPT4

1.8 T model

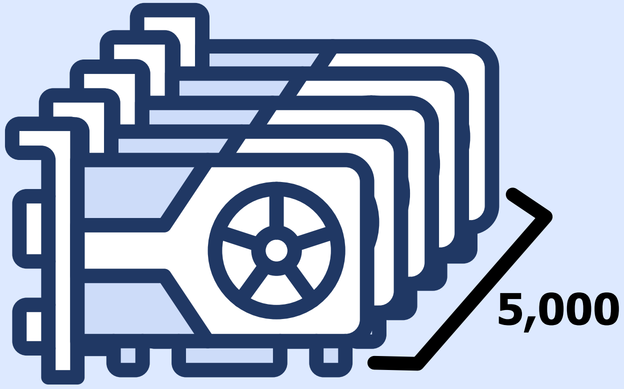
10,000 users

1,024 tokens

Computation

10^{19} FLOPs

5,000x

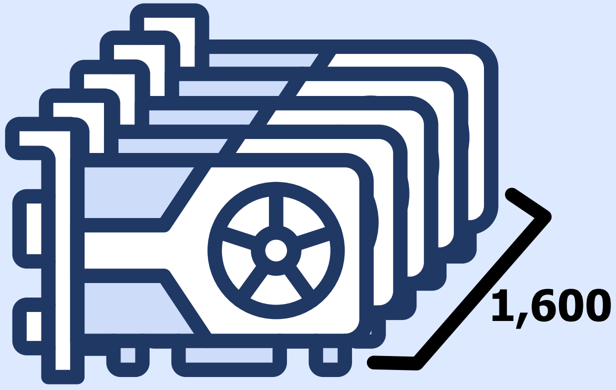


NVIDIA H100 GPUs

Memory

130 TB

1,600x

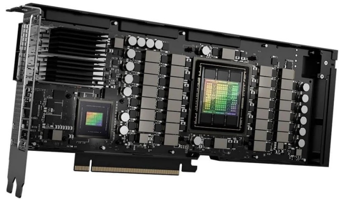


NVIDIA H100 GPUs

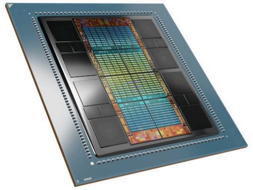
Limitations of Existing ML System Simulators

Lack of Support for Heterogeneous Hardware

GPU

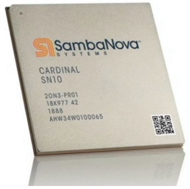


NVIDIA

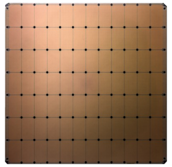


AMD

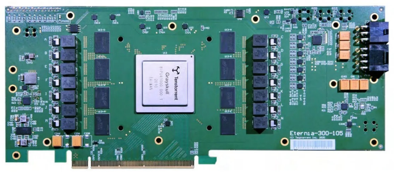
NPU



SambaNova SYSTEMS



cerebras



tenstorrent

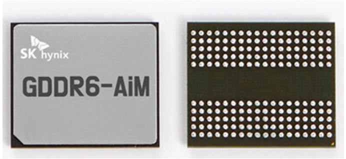
GRAPHCORE

rebellions_

FURIOSA

PIM

SAMSUNG



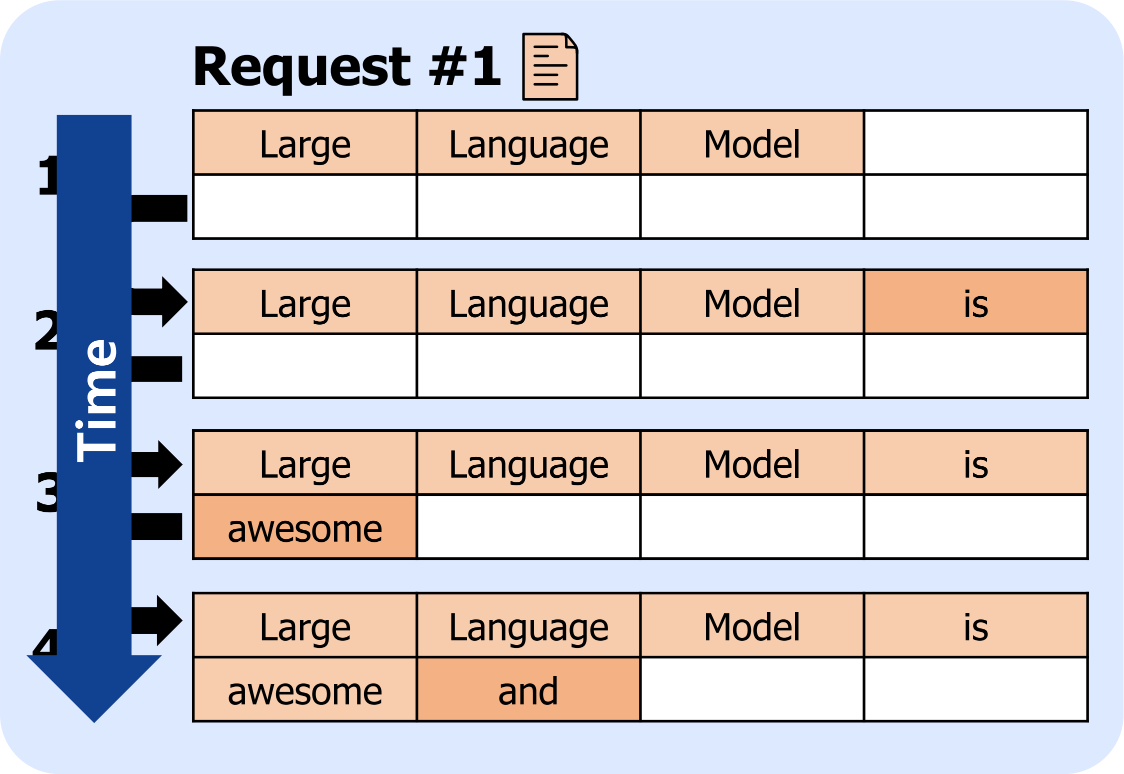
SK hynix

up mem

Limitations of Existing ML System Simulators

Lack of Support for Dynamically Changing Workload

LLM Inference Serving System



Request #1

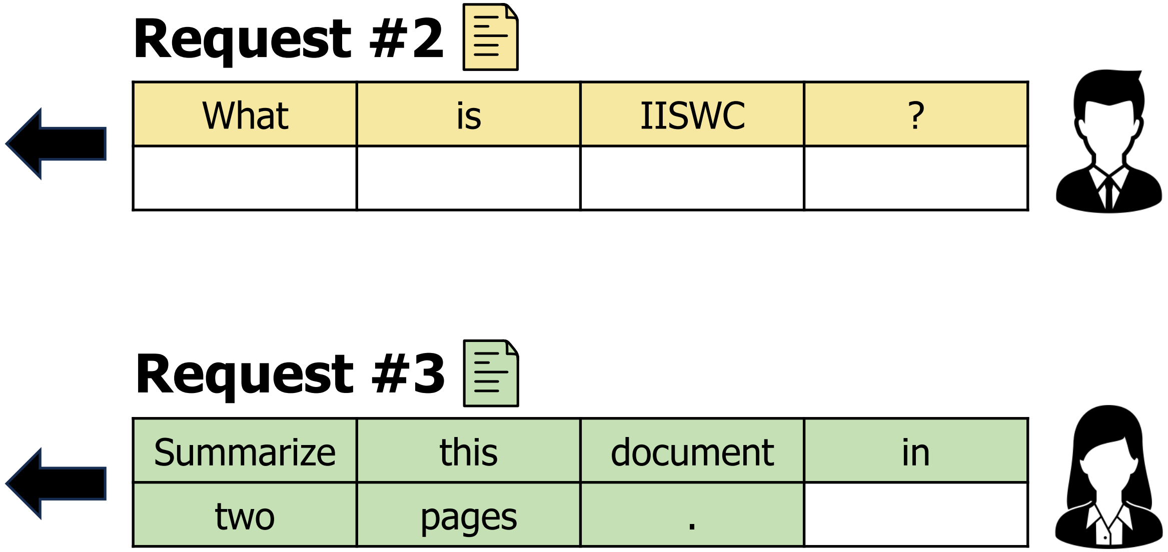
Large	Language	Model	

Large	Language	Model	is

Large	Language	Model	is
awesome			

Large	Language	Model	is
awesome	and		

Autoregressive Generation



Request #2

What	is	IISWC	?

Request #3

Summarize	this	document	in
two	pages	.	

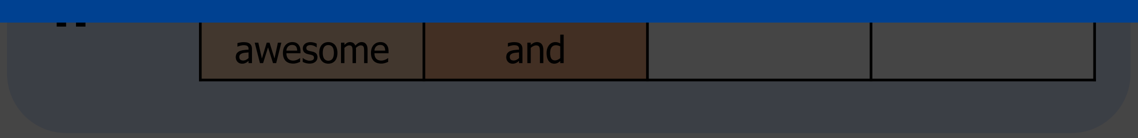
Request from Users at Random Time

Limitations of Existing ML System Simulators

Lack of Support for Dynamically Changing Workload

HW/SW Co-Simulation Infrastructure for
LLM Inference Serving

LLMServingSim



Autoregressive Generation

Request from Users at Random Time

Overview of LLMServingSim

Challenges

① Autoregressive LLM inference

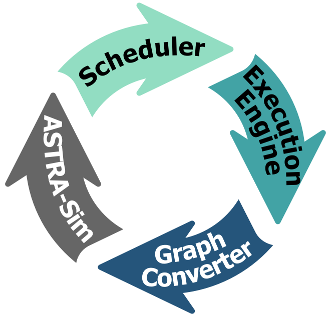
② LLM specific parallelism

③ Slow hardware simulation time

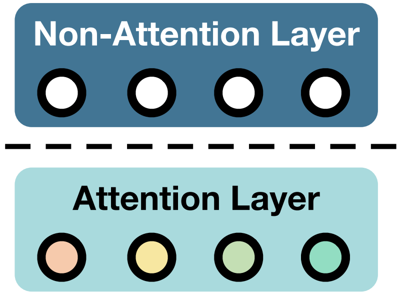
④ Heterogeneity Support

Solutions

① Iterative workflow

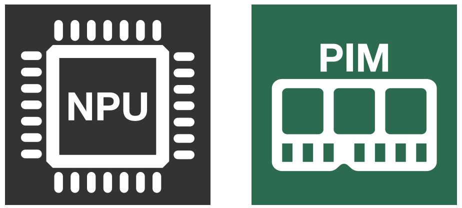


② Layer-Specific Processing



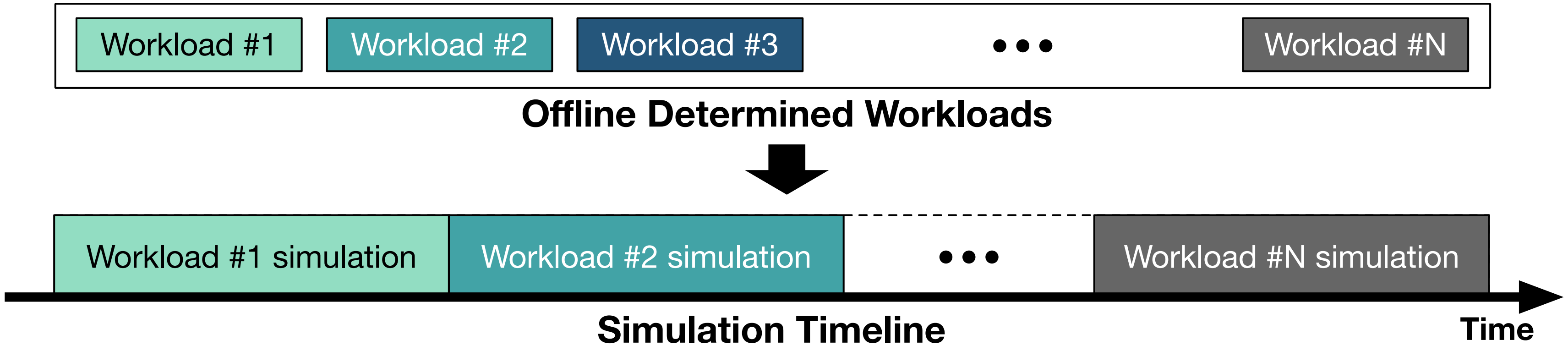
③ Computation Reuse

④ Heterogeneous System

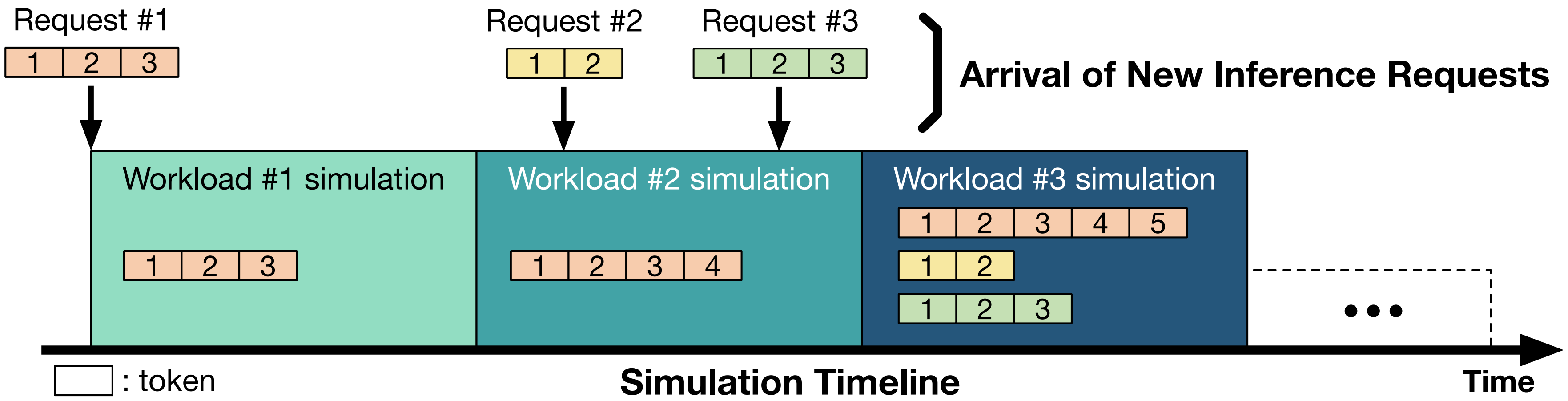


Challenge 1: Autoregressive LLM

- **Existing simulators run static workloads**



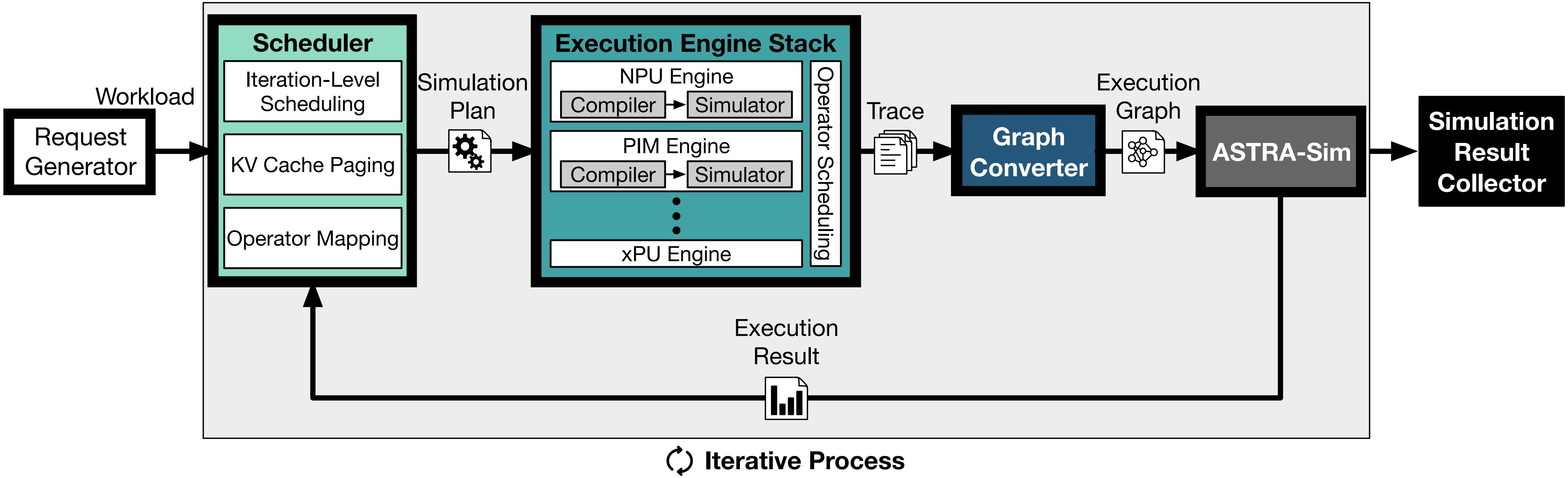
- **In LLM inference, workload dynamically changes at runtime**



Solution 1: Iterative Workflow

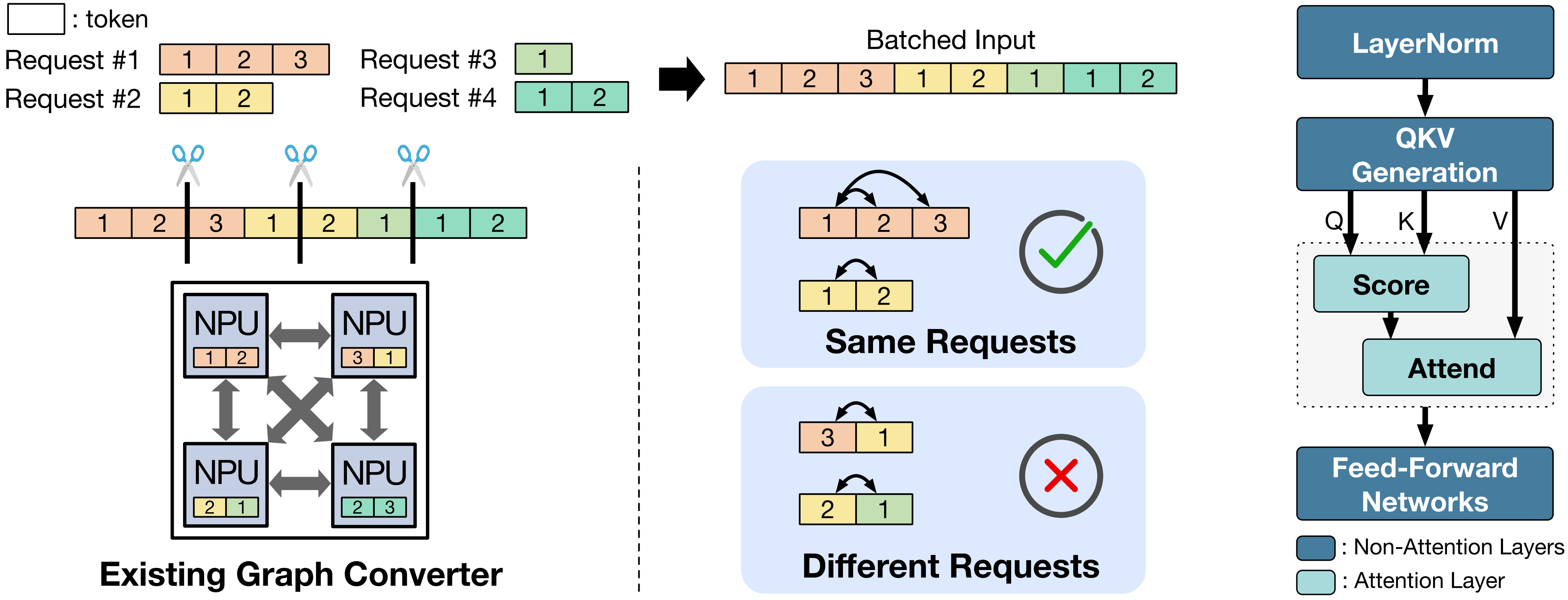
- LLMServingSim operates in an **iterative manner**

[illegible]



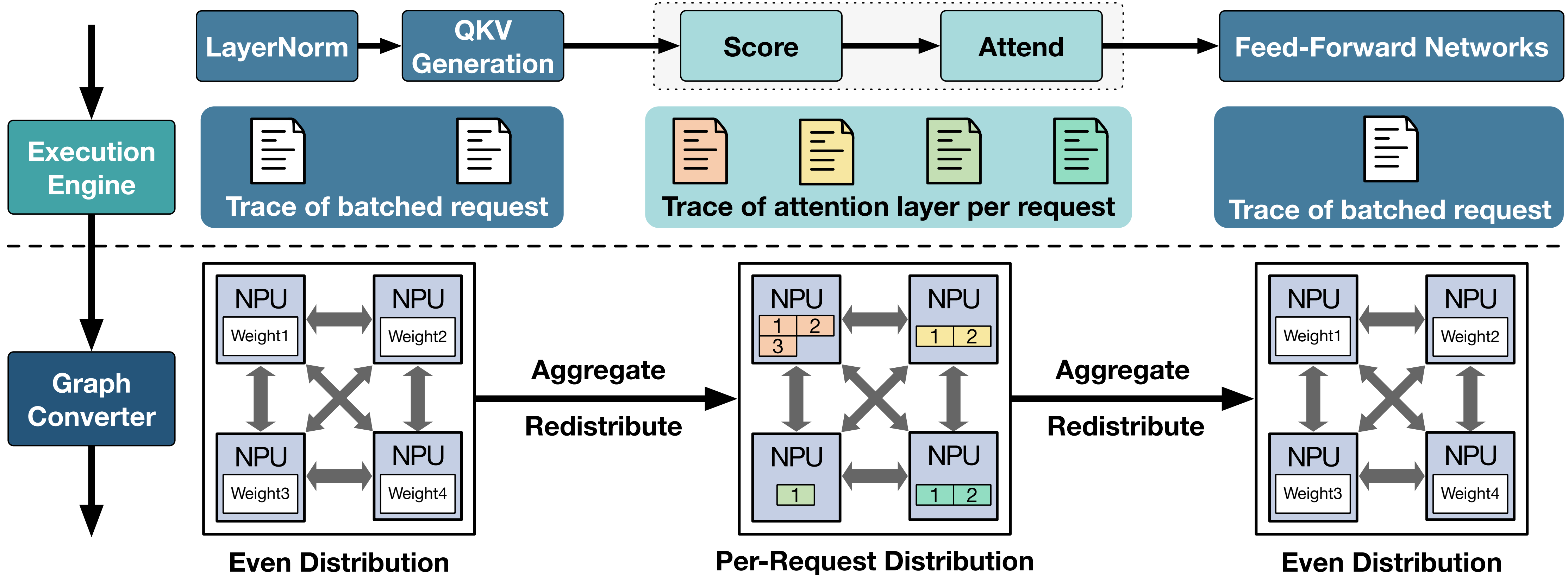
Challenge 2: LLM Specific Parallelism

- **Existing graph converter splits input evenly**
- **Attention should only be applied within the same request**



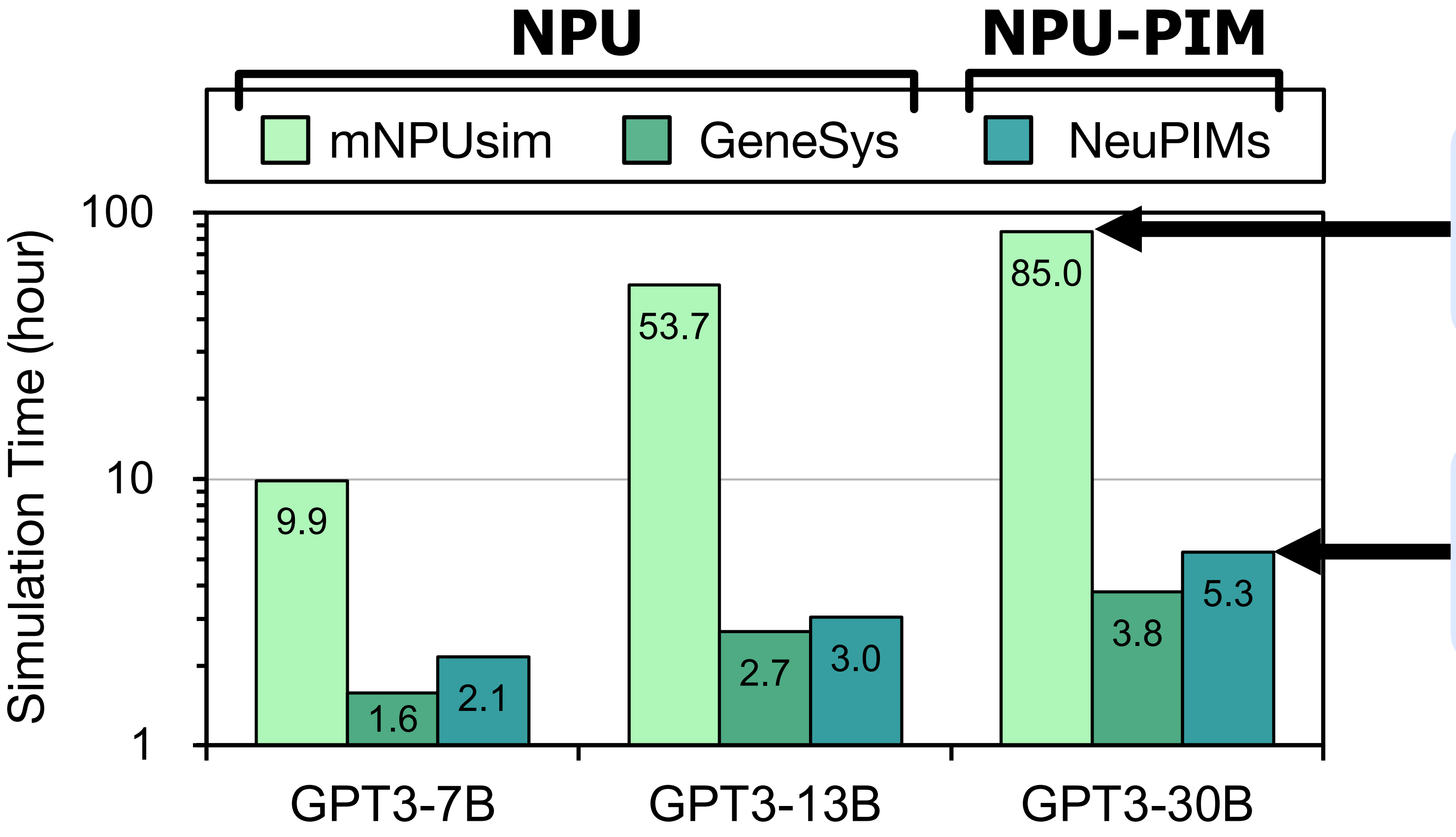
Solution 2: Layer-Specific Processing

- Execution Engine **runs different operators** according to the layer type
- Graph Converter **distributes operators** according to the layer type



Challenge 3: Slow HW Simulation Time

- **Simulation time of other LLM hardware simulators**
- **Batch: 32, Sequence length: 512**



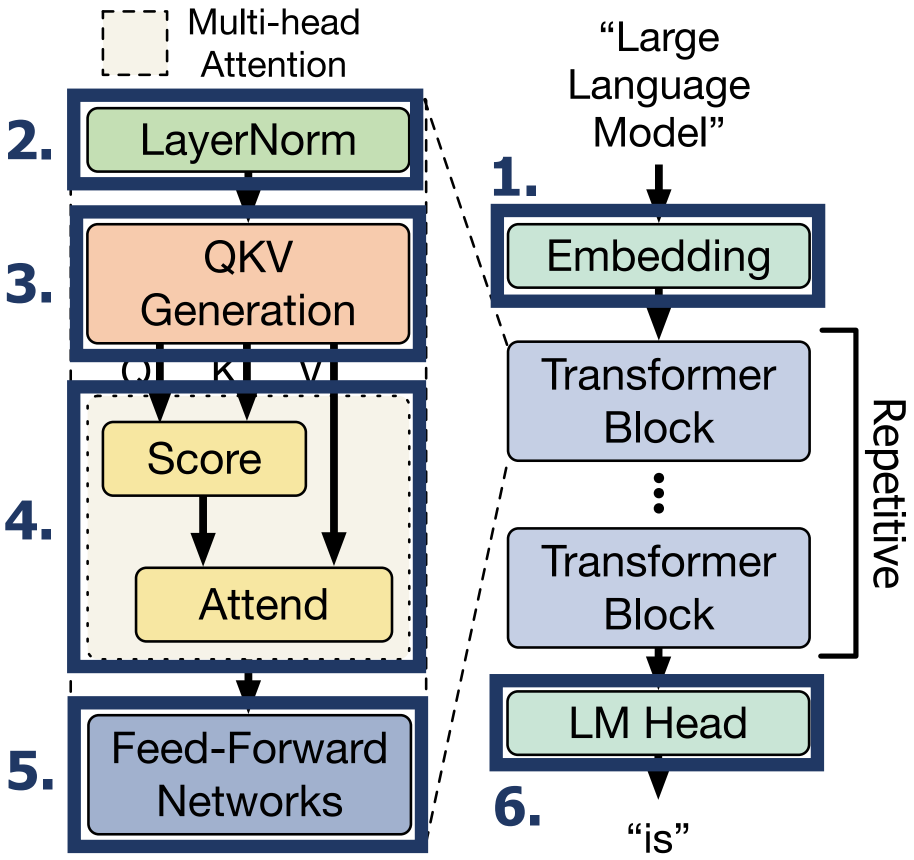
More than **85 hours** for 1 iteration

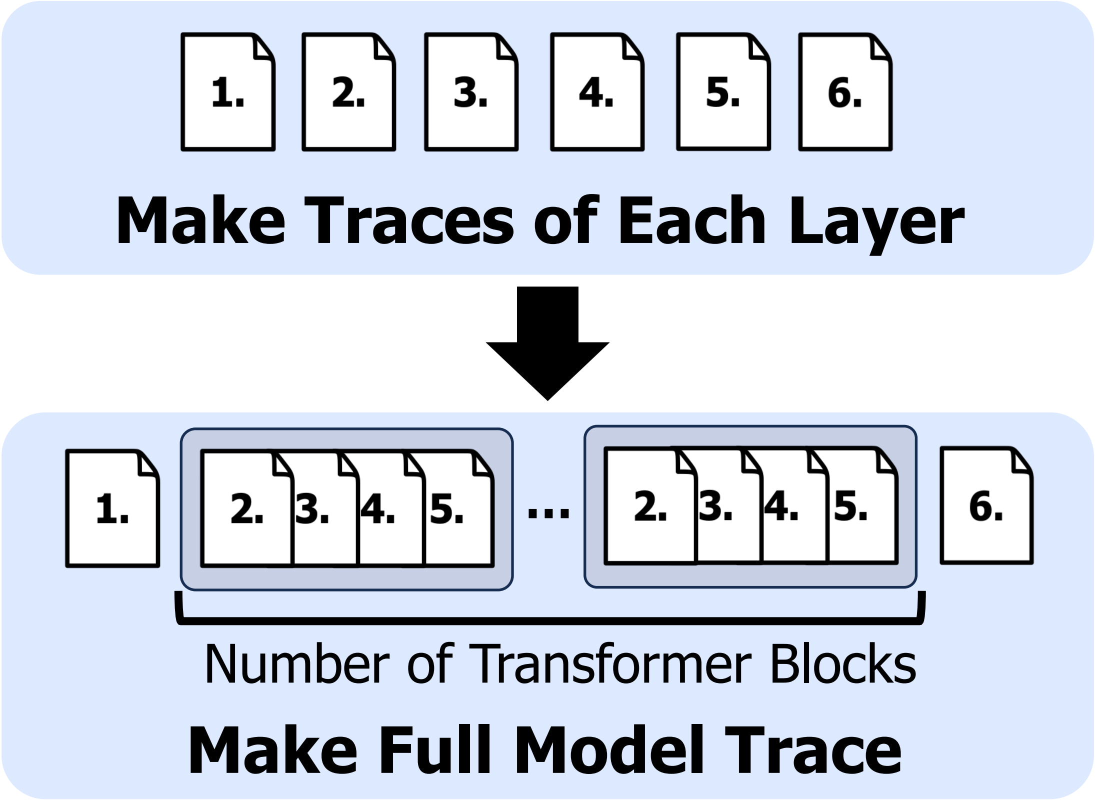
More than **5 hours** for 1 iteration

Solution 3: Computation Reuse

Leveraging the Repetitive Structure of LLM

- **Split** the model into 6 layers and compile it once
- **Combine** traces to make full model

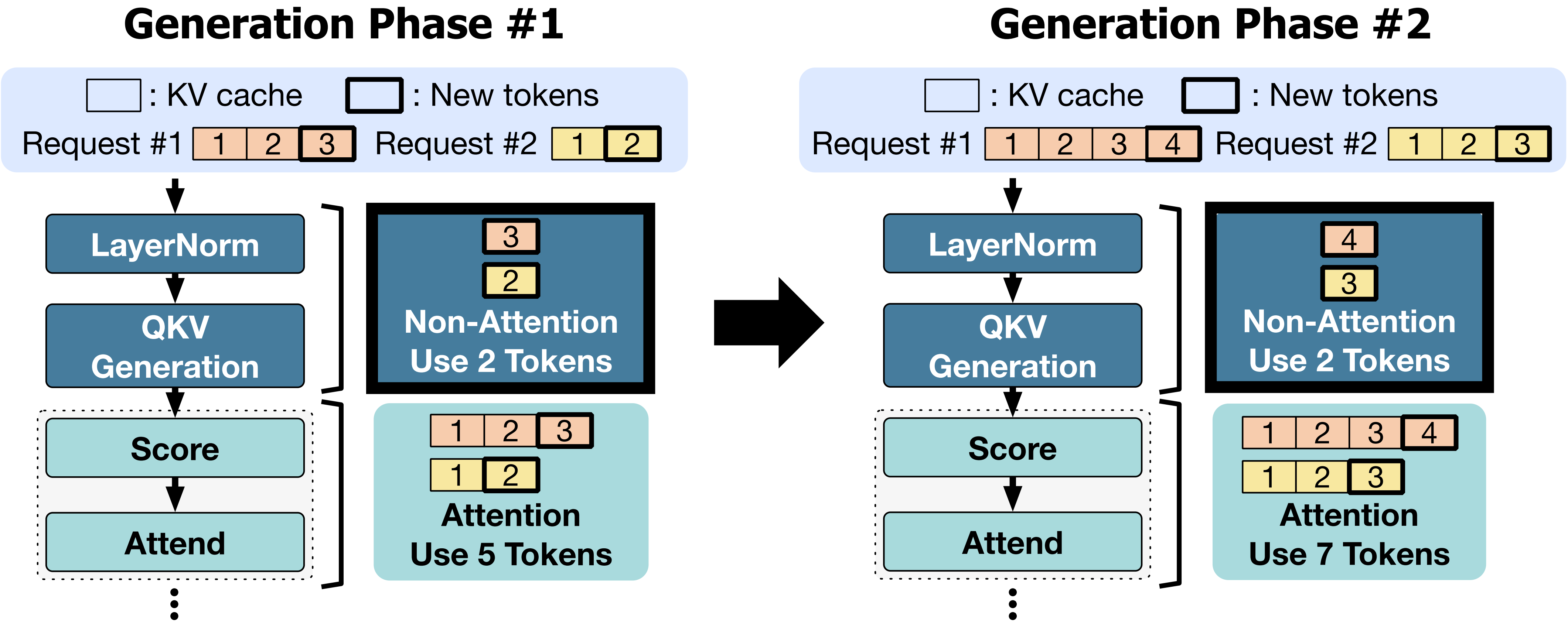




Solution 3: Computation Reuse

Leveraging the Locality of LLM Inference

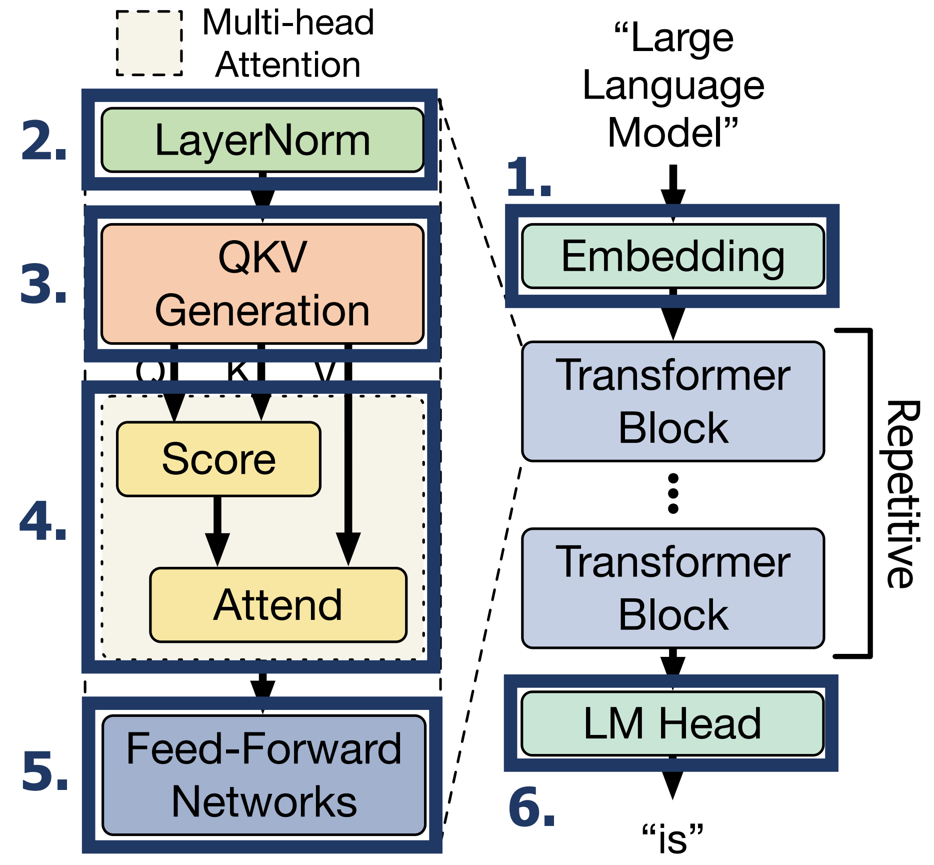
- **Non-attention layers use same number of tokens each iteration**
- **Generation phase occurs more frequently than initiation phase**

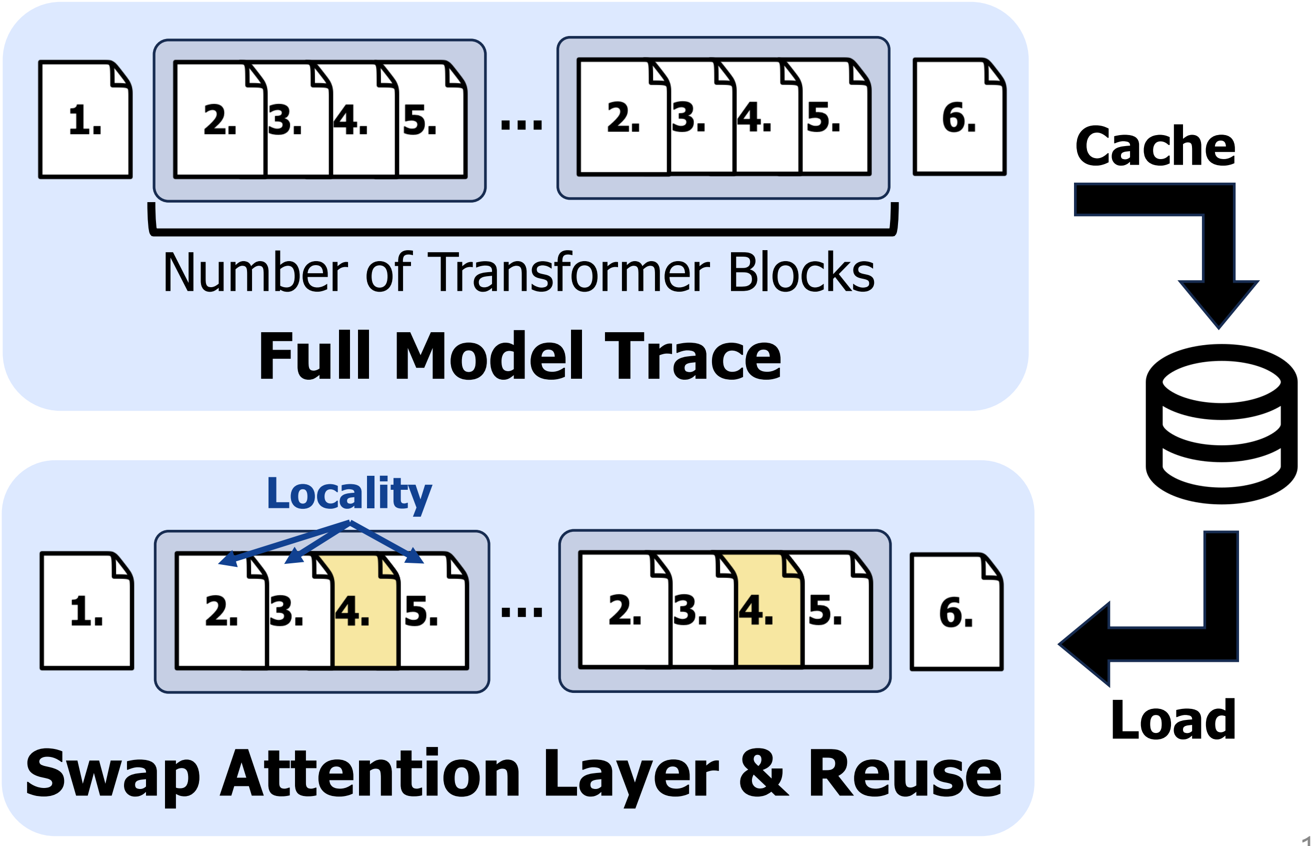


Solution 3: Computation Reuse

Leveraging the Locality of LLM Inference

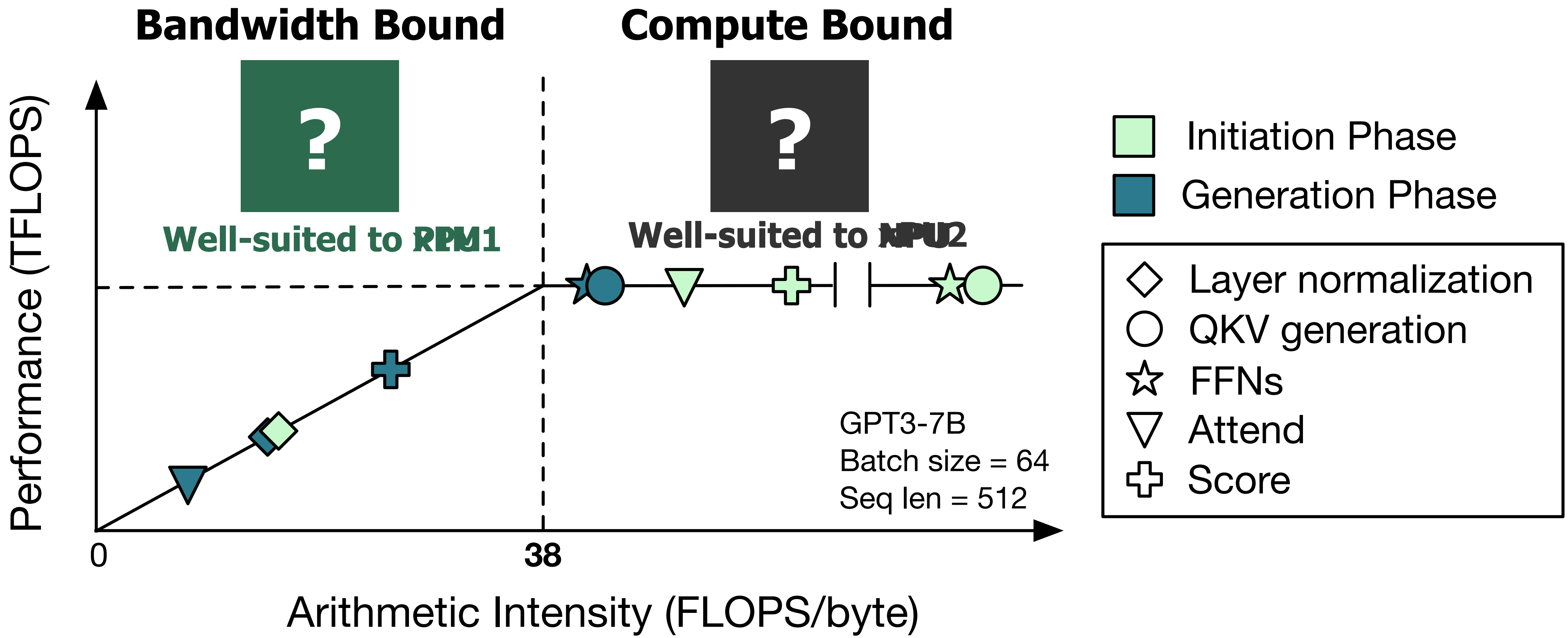
- **Reuse the model trace by swapping out the attention layer**
- **Eliminates most time-consuming hardware simulation**





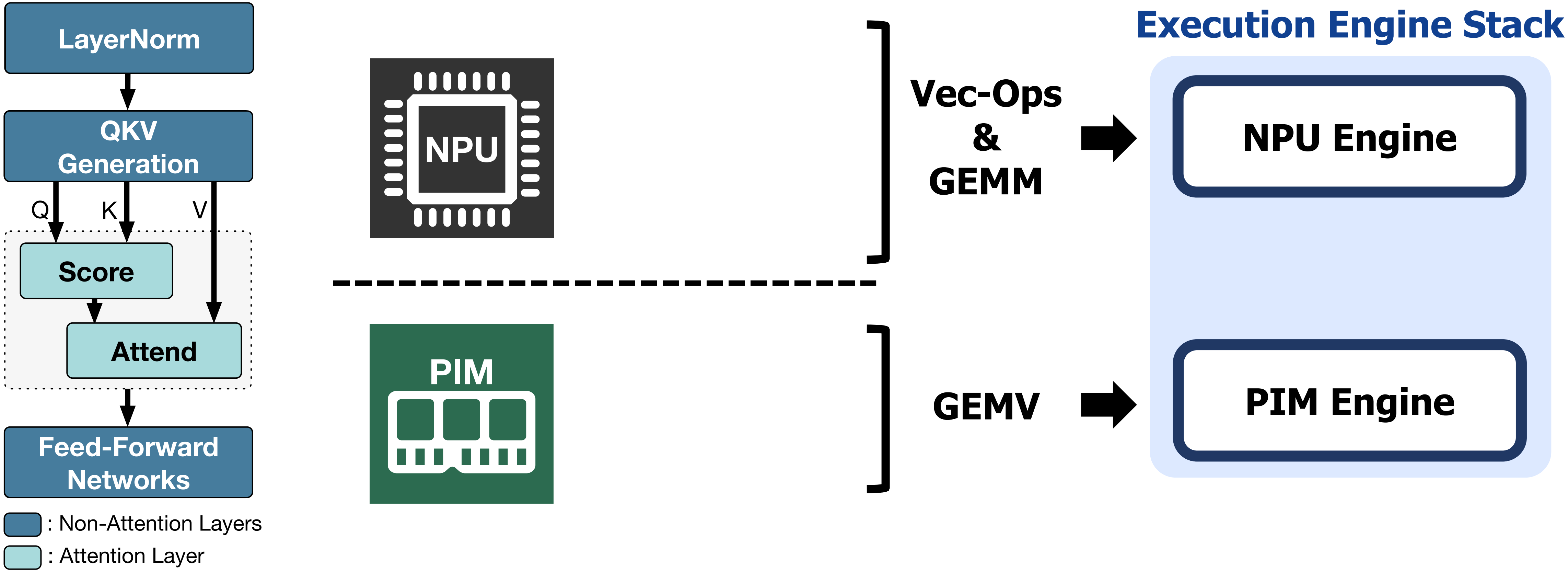
Challenge 4: Heterogeneity Support

- **Heterogeneous accelerators with different characteristics**



Solution 4: Operator Mapping

- Map each operator to **specific hardware type**
- Each execution engine **compiles and simulates** mapped operators



LayerNorm

QKV Generation

Q K V

Score

Attend

Feed-Forward Networks

▇ : Non-Attention Layers

▇ : Attention Layer

NPU

PIM

Vec-Ops & GEMM

GEMV

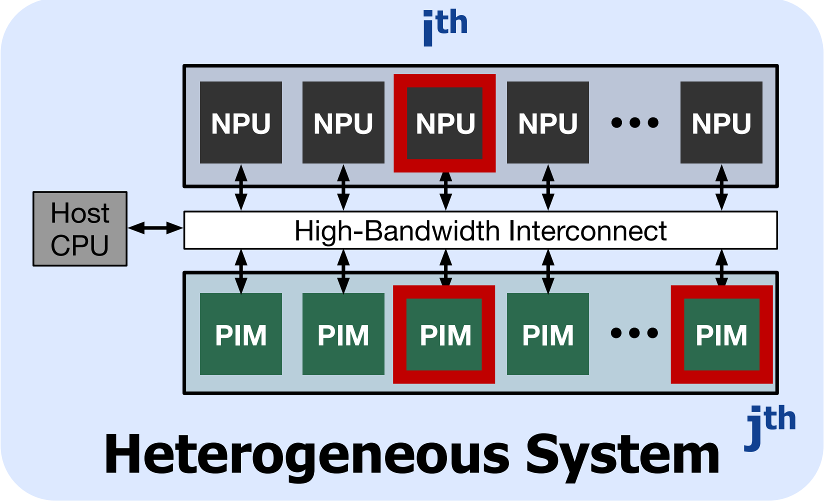
Execution Engine Stack

NPU Engine

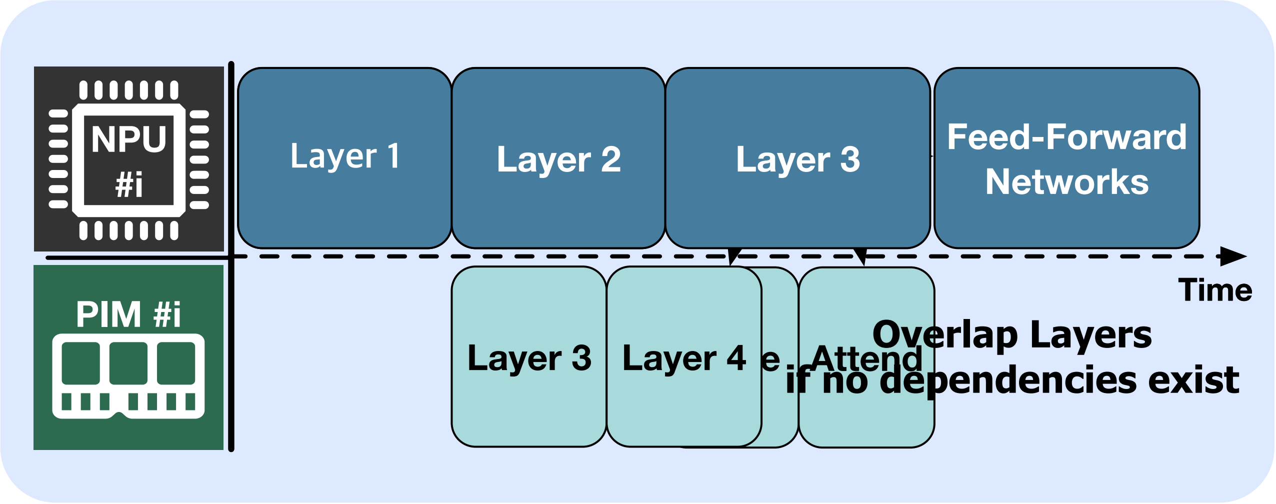
PIM Engine

Solution 4: Operator Scheduling

- Schedule each operator to **specific hardware**
- Scheduling algorithm based on **system topology** and **dependencies**
- **Flexible** algorithm configuration



System Topology



Layer Dependencies

Evaluation Methodology

- **Real-System Baseline**
  - vLLM Framework with 4 NVIDIA RTX 3090 GPUs

- **Simulator Baseline**
  - mNPUsim[1]
  - GeneSys[1] - Used in LLMServingSim
  - NeuPIMs[2]

[1] NPU simulator
[2] NPU-PIM simulator

- **Dataset**
  - ShareGPT
  - Alpaca

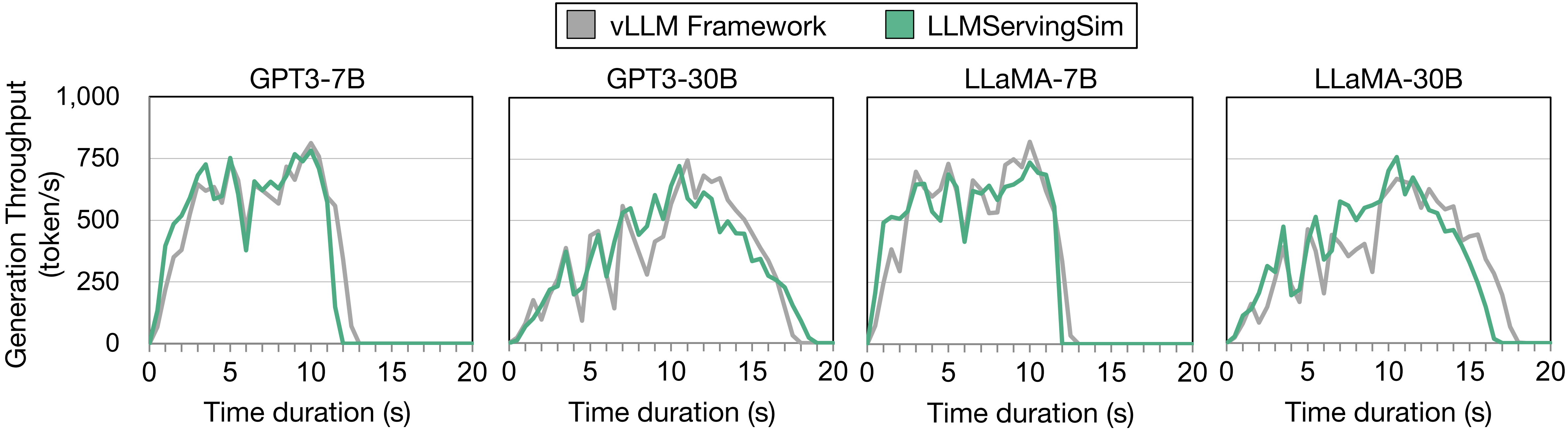
NPU Configuration	
Systolic Array	128x128
Vector Unit	128x1
Frequency	1GHz
Memory Capacity	24GB
Internal Bandwidth	936GB/s

PIM Configuration	
Banks / Bankgroup	4
Banks / Channel	32
Frequency	1GHz
Memory Capacity	32GB
Internal Bandwidth	1TB/s

Inter-device Link Configuration	
Bandwdith	64GB/s
Latency	100ns

Validation of LLMServingSim

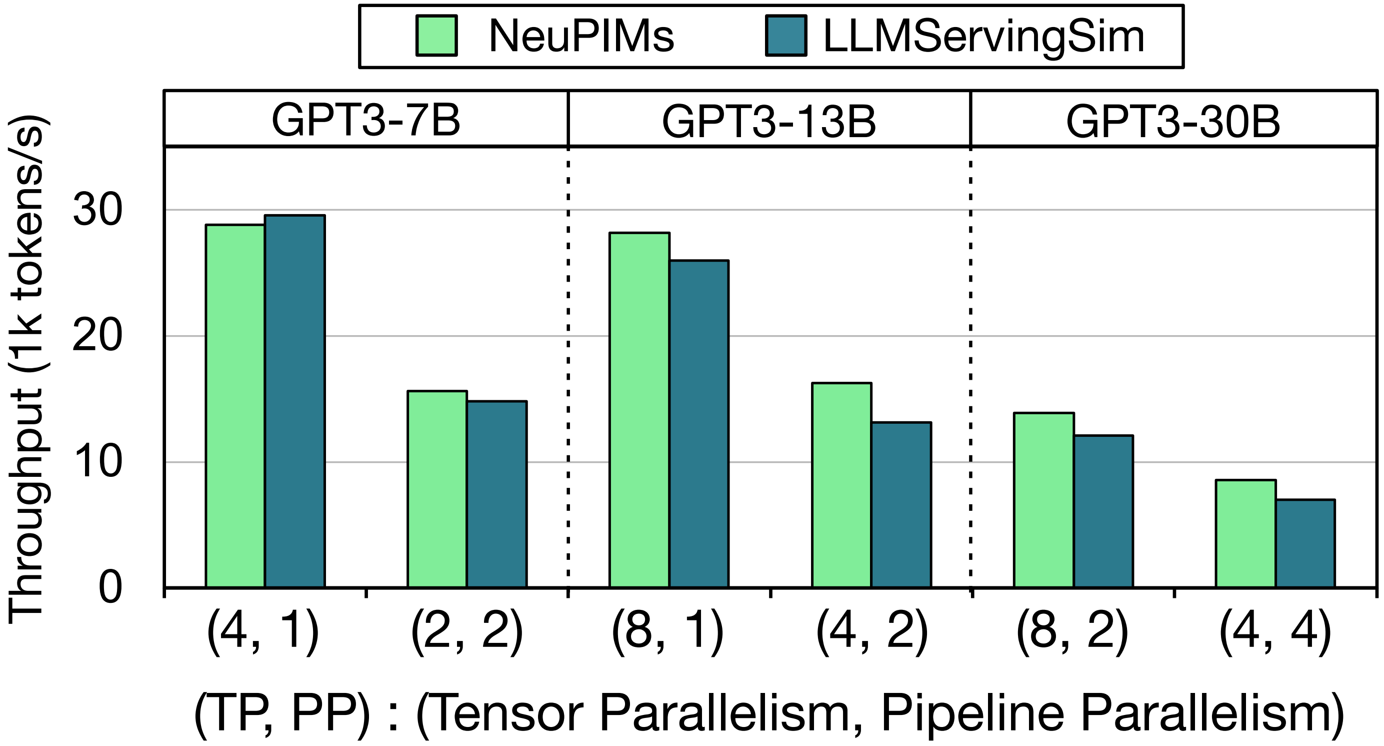
NPU Homogeneous System



- **High similarity between real-system and LLMServingSim**
- **Average error rate 14.7%**

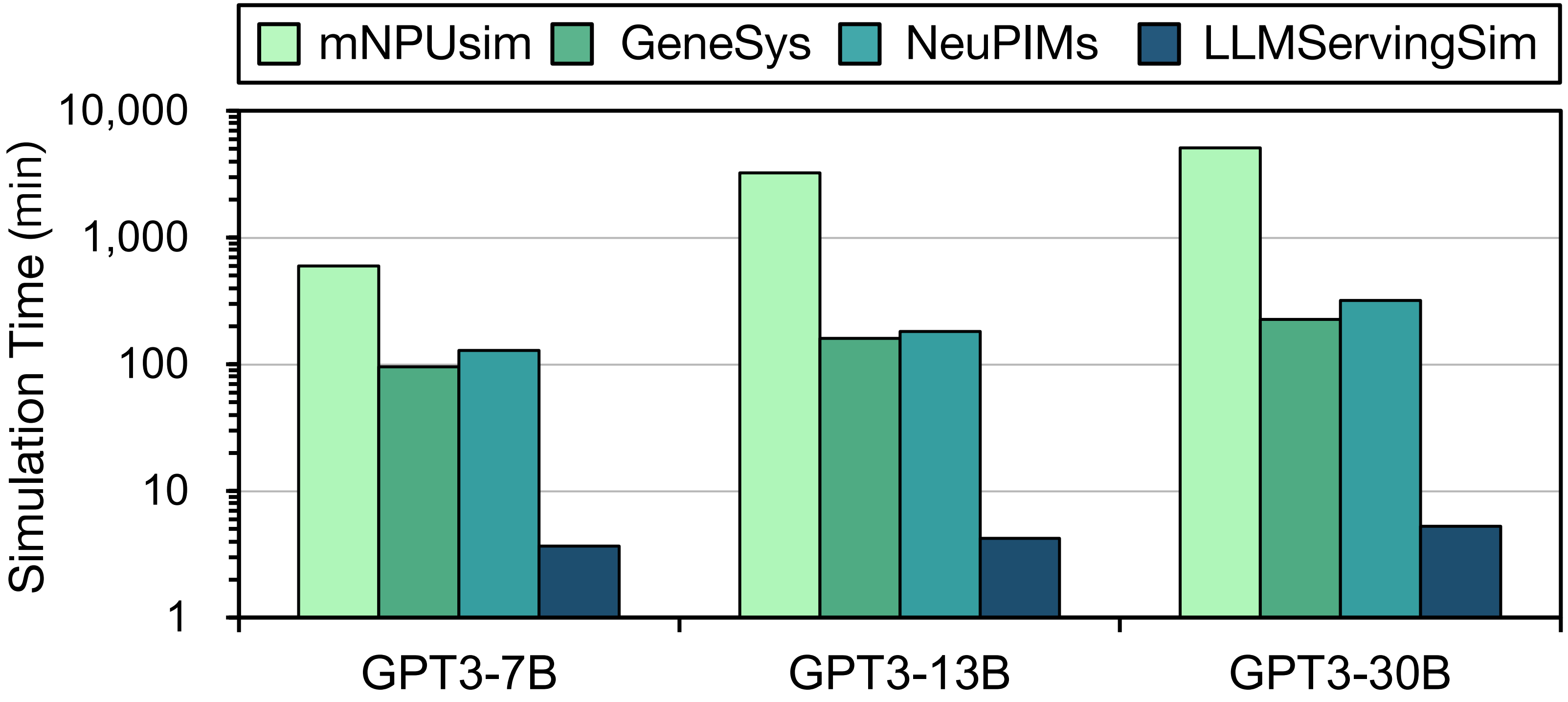
Validation of LLMServingSim

NPU-PIM Heterogeneous System



- **High similarity between NPU-PIM simulator and LLMSergvingSim**
- **Average error rate 8.88%**

Simulation Time Comparison



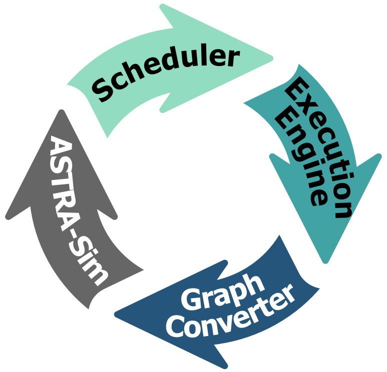
- **491.0x, 34.7x, 45.0x faster than mNPUsim, GeneSys, NeuPIMs**
- **LLMServingSim achieved fast simulation through computation reuse**

Conclusion

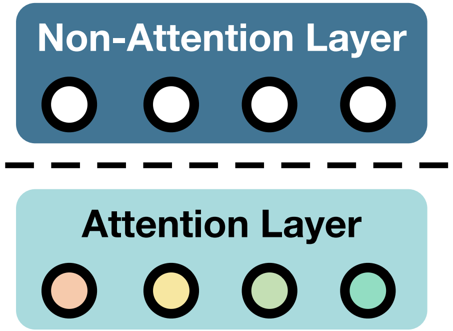
Our simulator code is available
https://github.com/casys-kaist/LLMServingSim

- **LLMServingSim**
  - HW/SW co-simulation infrastructure for LLM inference serving
- **Contributions**
  - Iterative workflow for autoregressive LLM inference
  - Layer-specific processing for LLM specific parallelism
  - Computation reuse to reduce simulation time
  - Heterogeneous accelerators support with easy integration

Iterative Workflow



Layer-Specific Processing



Computation Reuse

Heterogeneous System

NPU

PIM

Performance

14.7%
Error Rate

91.5x
Faster Simulation