PAPI: Exploiting Dynamic Parallelism in Large Language Model Decoding with a Processing-In-Memory-Enabled Computing System

Yintao He Haiyu Mao Christina Giannoula
Mohammad Sadrosadati Juan Gómez-Luna Huawei Li
Xiaowei Li Ying Wang Onur Mutlu

ASPLOS 2025

SAFARI

Executive Summary

Observation: Large Language Model (LLM) decoding kernels have **different and dynamically changed** computation and memory bandwidth demands at runtime

Problem: The existing designs have two shortcomings:

- **Static scheduling** that fails to dynamically cater to the changing kernel demands
- **Support only one type of Processing-In-Memory (PIM) device** with a certain computation throughput and memory bandwidth capability

Goal: Design a **heterogeneous system** that caters to different and dynamically changing computation and memory demands

Key Idea: Enable **online dynamic task scheduling** on a heterogeneous architecture via online identification of kernel properties in LLM decoding

Key techniques: A new computing system called **PAPI** with

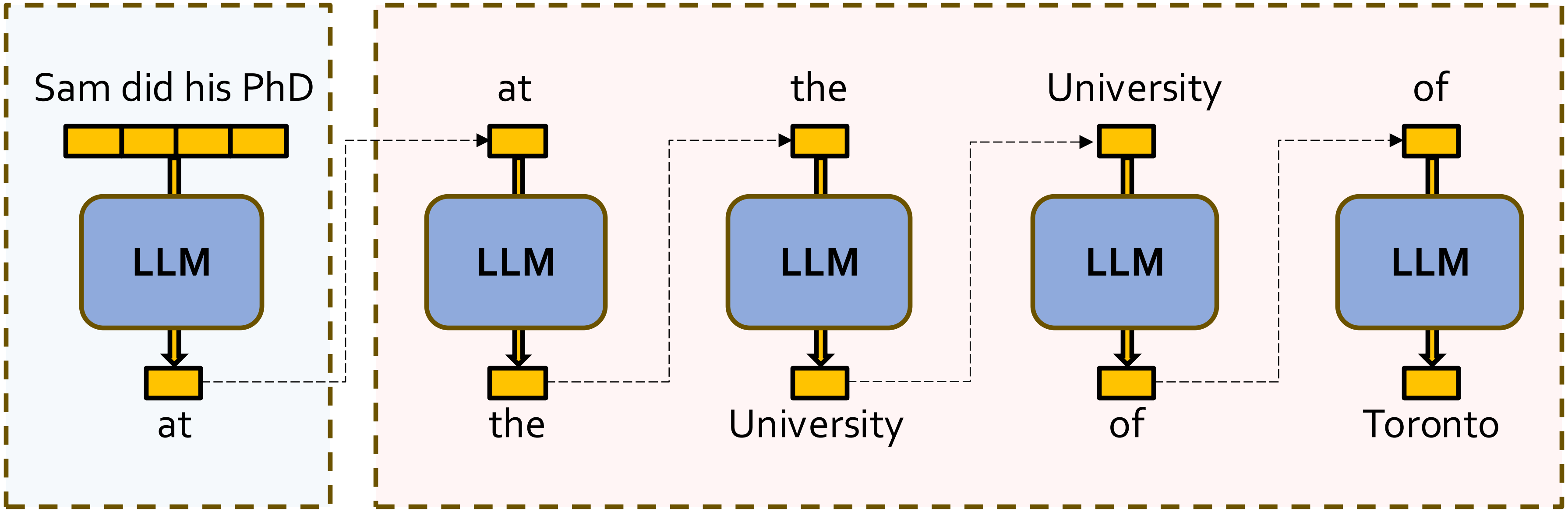
- **Dynamic LLM kernel scheduling** to the most suitable hardware units at runtime
- **Hybrid PIM units** to meet the diverse LLM kernel demands

Key Results: PAPI outperforms a state-of-the-art PIM-enabled LLM computing system and a pure PIM system by **1.8X** and **11.1X**, respectively

Outline

LLM Inference

An example:



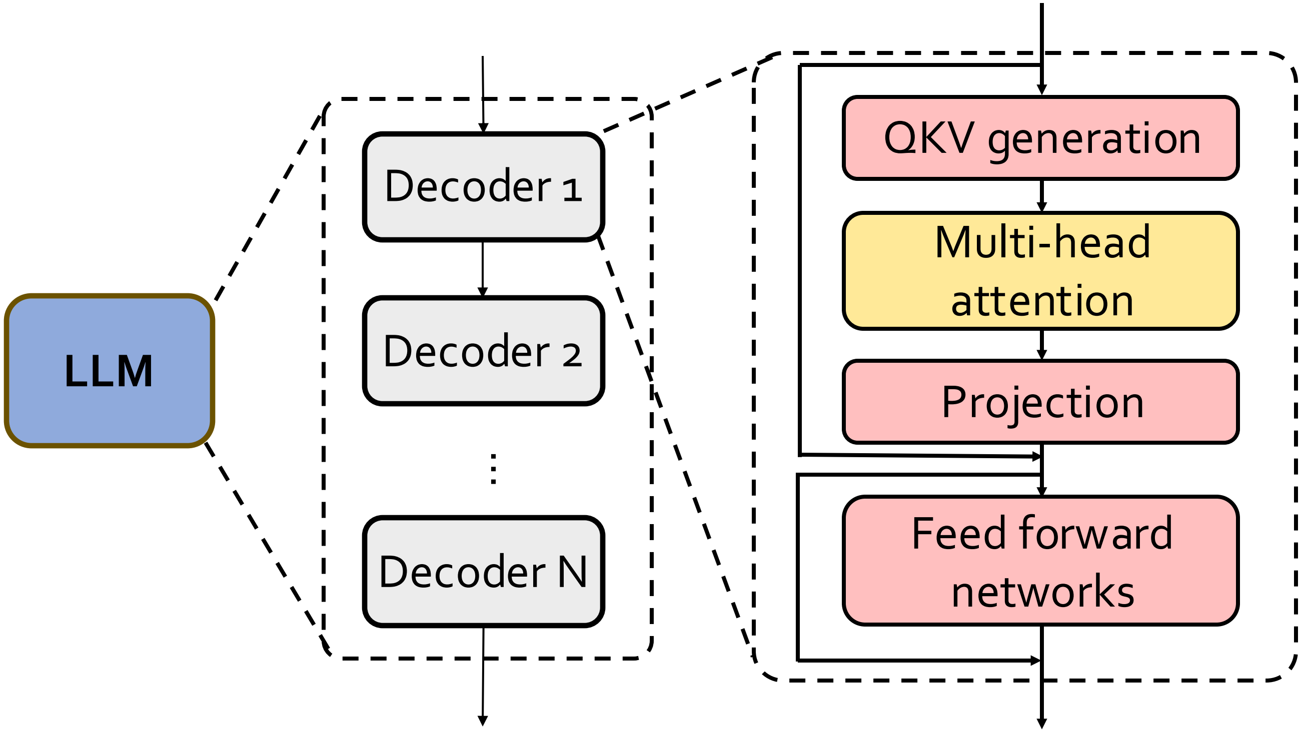
Prefilling

(Encodes contextual information from the input in parallel)

Decoding

(Generates output tokens **in serial / parallel**)

LLM Structure



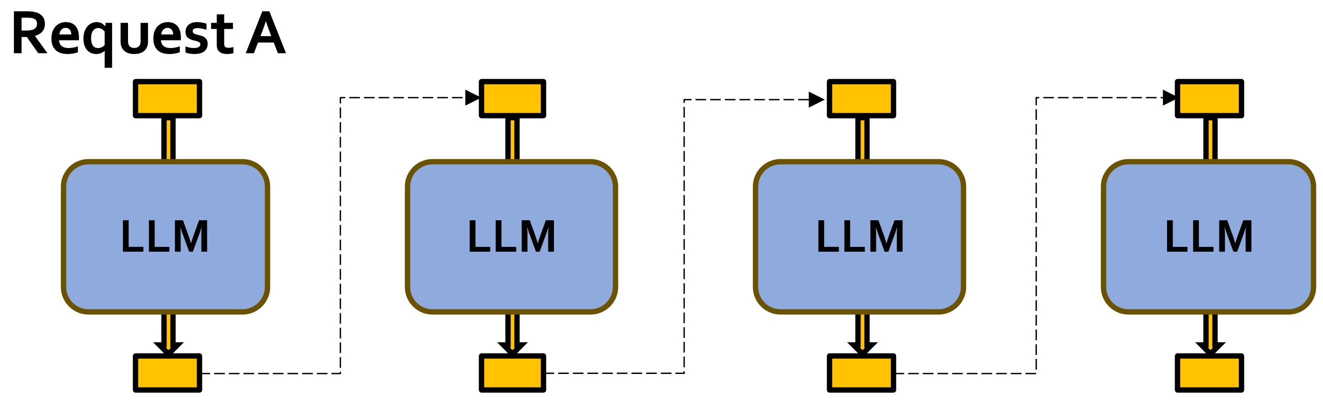
Attention kernels

- Encoded from input tokens
- Different data across requests

Fully-connected (FC) kernels

- Pretrained by LLM training
- Used for all token generation

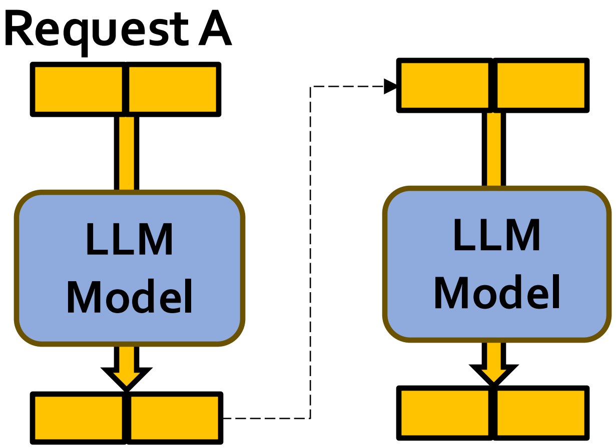
Serial Decoding



- **Low hardware utilization**
- **Low throughput**

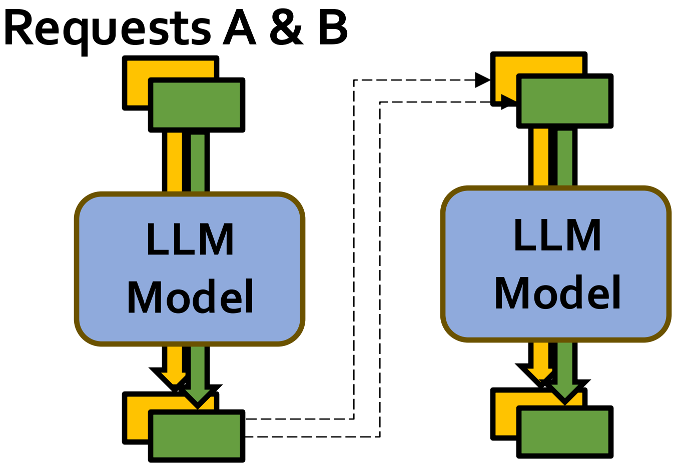
Parallel Decoding

Decode tokens of a request in parallel



Token-Level Parallelism (TLP)

Decode different requests in parallel



Request-Level Parallelism (RLP)

- Higher hardware utilization
- Higher throughput

Do TLP and RLP benefit all kernels in LLM decoding?

Outline

Key Observations

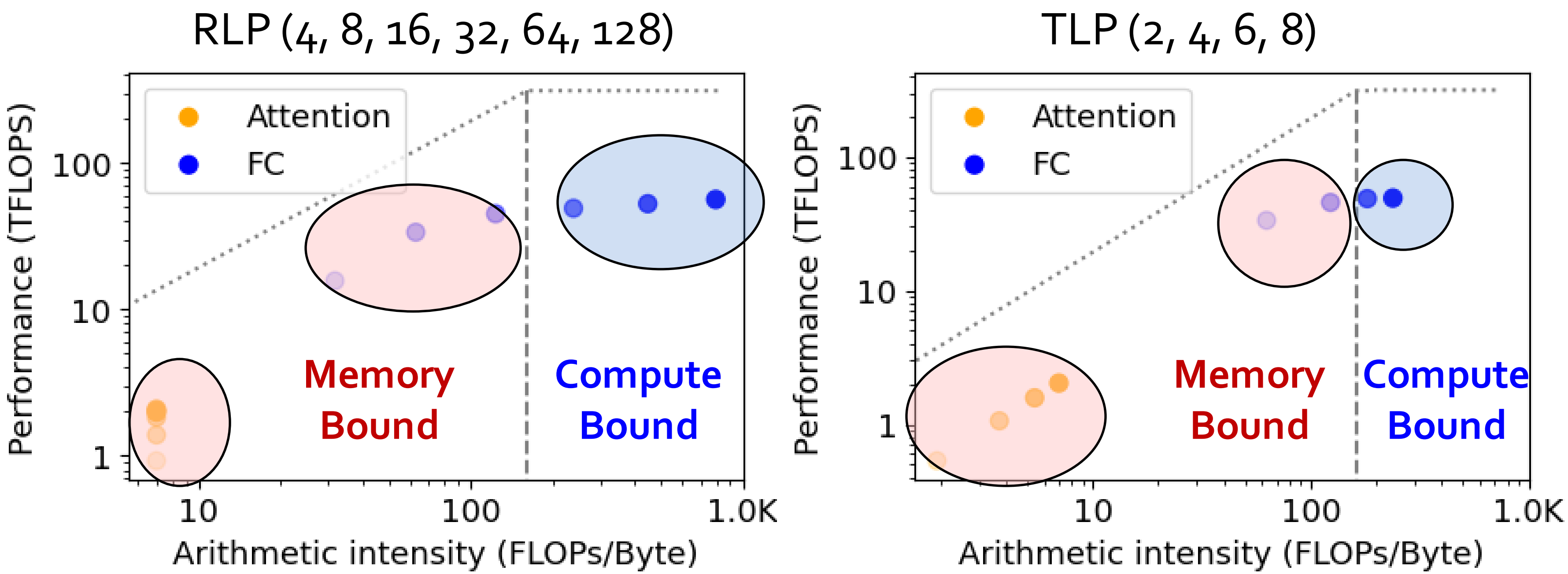
1 There are **varying computation and memory bandwidth demands** across **different RLP & TLP configurations**

2 The **memory-bound kernels** exhibit **various computation demands** depending on the kernel type

3 LLM kernels have **dynamic computation demands** at runtime

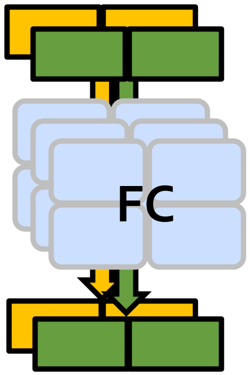
Varying Computation and Memory Bandwidth Demands (i)

The roofline model of LLM kernels with **six RLP and four TLP configurations** on a NVIDIA A100 **GPU system**:



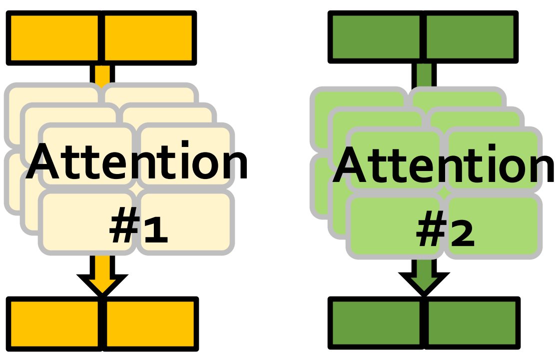
There are varying computation and memory bandwidth demands across different RLP & TLP configurations

The Reason for Different Demands



- **FC kernels** benefit from **RLP & TLP**

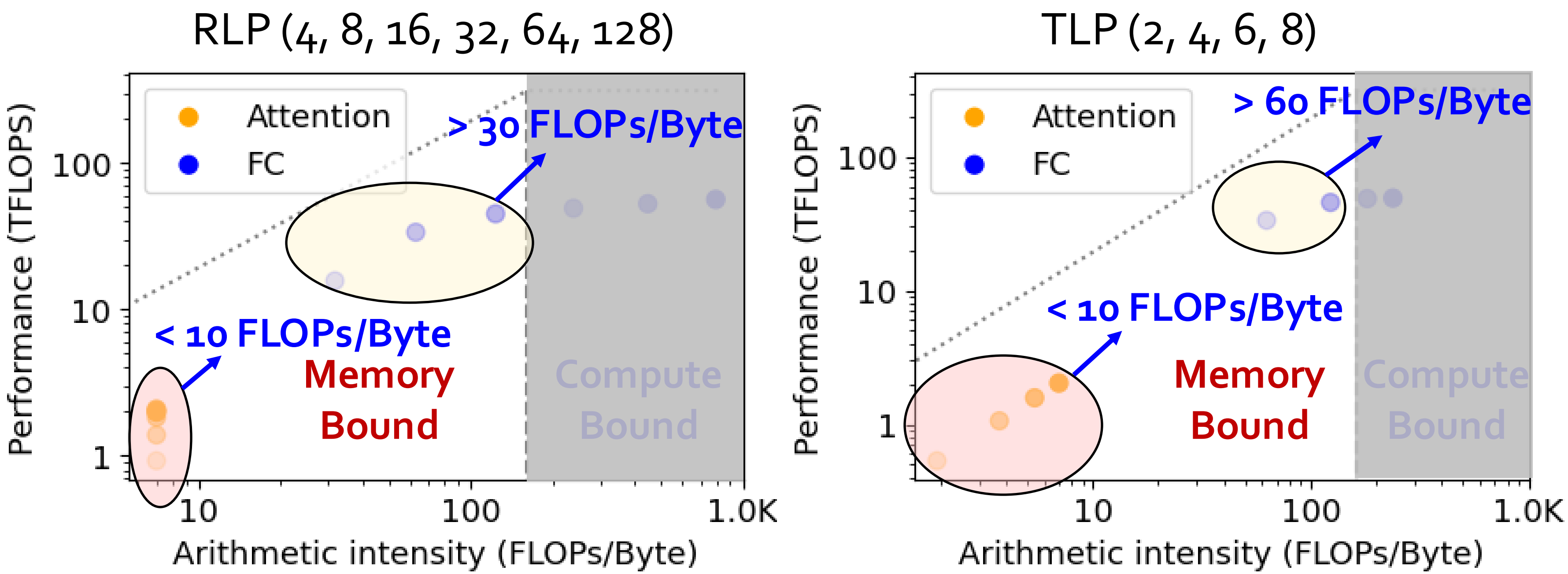
Compute-Bound



- **Attention kernels** benefit from **TLP**
- **TLP** is usually **much smaller** than RLP

Memory-Bound

Varying Computation and Memory Bandwidth Demands (ii)

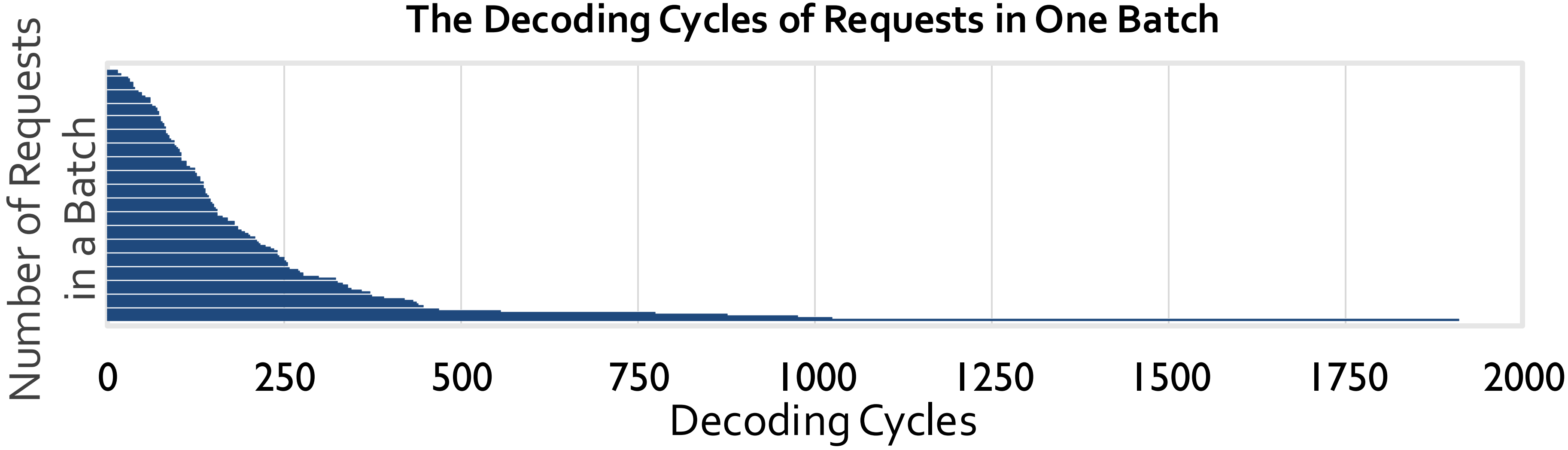


The memory-bound kernels exhibit various computation demands depending on the kernel type

Dynamic Parallelism Levels

- **Parallelism levels** (RLP & TLP) **vary dynamically** in real-world scenarios
  - Request-level parallelism (RLP) **decreases at runtime** when using static batching

The Decoding Cycles of Requests in One Batch



In the Paper: Analysis of Dynamic Parallelism Levels

- **Initial RLP:**
  - Service level objective
  - Memory capacity limits
  - Dynamic batching

- **Runtime RLP:**
  - Static batching
  - Mixed continuous batching

- **TLP:**
  - Speculative decoding

LLM kernels have **dynamic computation demands** at runtime

In the Paper: Analysis of Dynamic Parallelism Levels

PAPI: Exploiting Dynamic Parallelism in Large Language Model Decoding with a Processing-In-Memory-Enabled Computing System

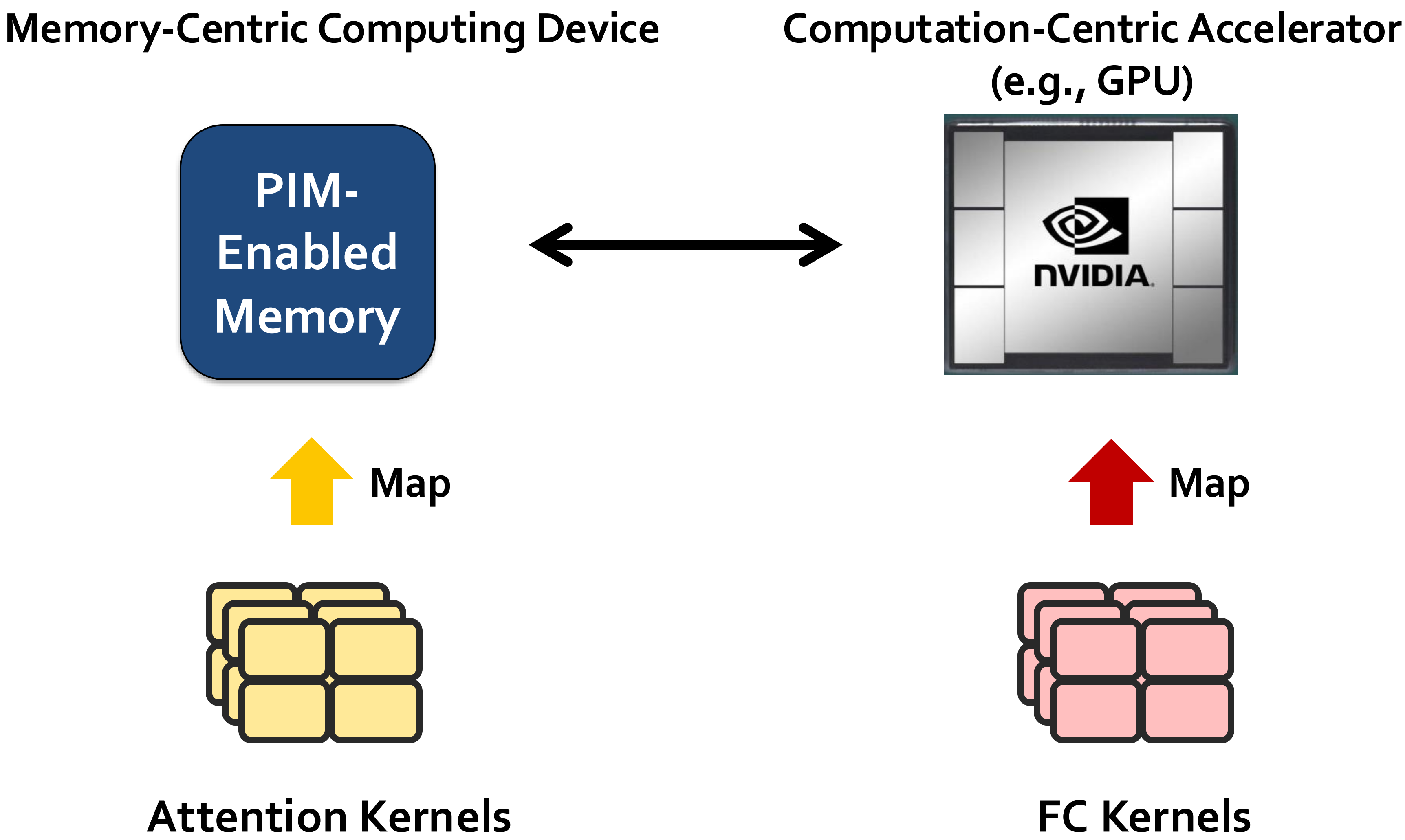
Yintao He[1,2] Haiyu Mao[3,4] Christina Giannoula[5,6,4] Mohammad Sadrosadati[4]
Juan Gómez-Luna[7] Huawei Li[1,2] Xiaowei Li[1,2] Ying Wang[1] Onur Mutlu[4]
[1]SKLP, Institute of Computing Technology, CAS [2]University of Chinese Academy of Sciences [3] King's College London
[4]ETH Zürich [5]University of Toronto [6]Vector Institute [7] NVIDIA

- TLP:
  - Speculative Decoding

https://arxiv.org/pdf/2502.15470

LLM kernels has dynan… on demands at runtime

The State-of-the-Art Approach



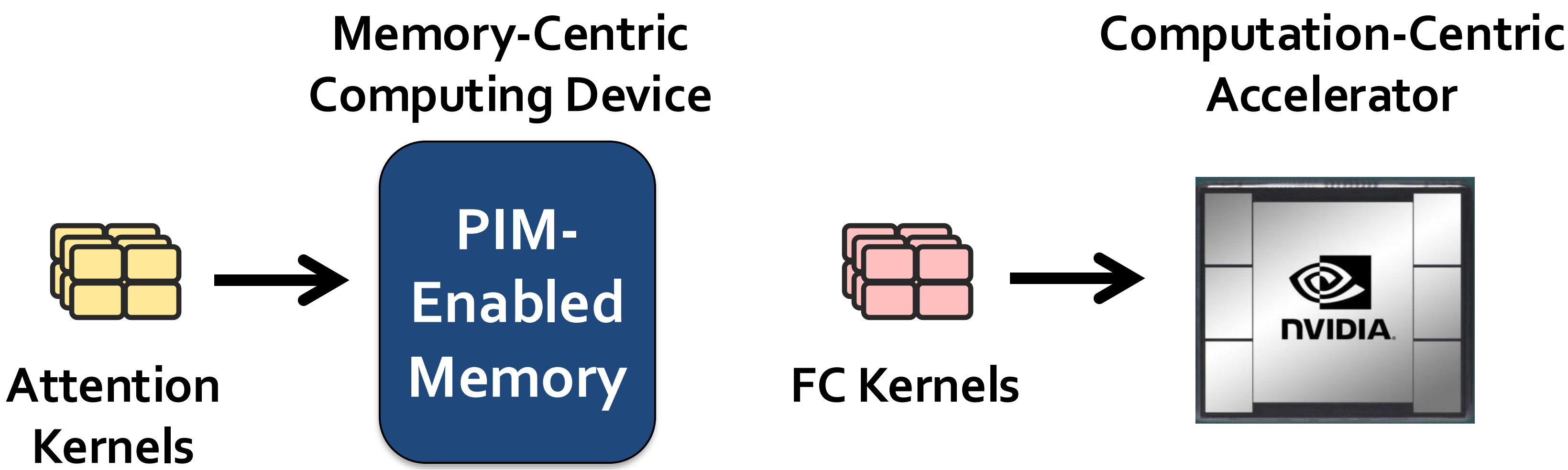
Major Shortcomings

1 **Static scheduling** leads to **sub-optimal** performance across **different parallelism levels**

2 The approach supports **only one type of PIM device** with a **certain computation and memory bandwidth capability**

(1) Static Scheduling (i)

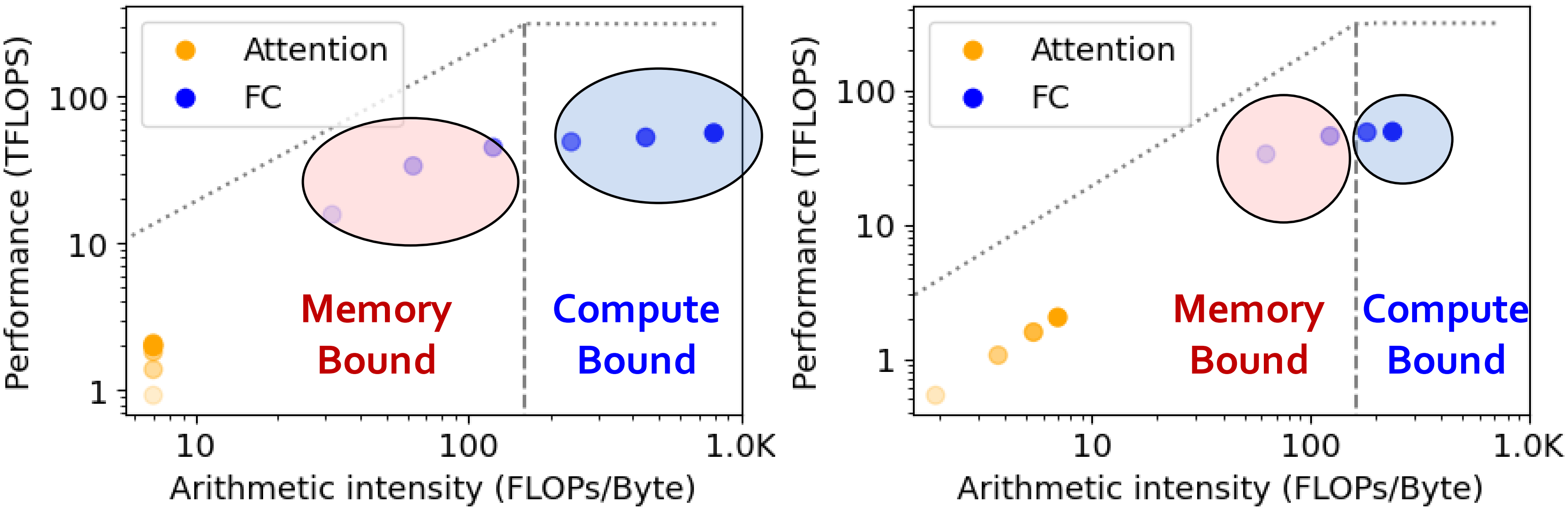
The state of the art approach typically take a static scheduling:



(1) Static Scheduling (ii)

- Static scheduling works **well** for **memory-bound attention kernels**
- Static scheduling **fails** for **FC kernels** that switch between being **compute-bound or memory-bound**

RLP (4, 8, 16, 32, 64, 128) TLP (2, 4, 6, 8)



Static scheduling leads to sub-optimal performance of FC kernels across different parallelism levels

(2) One Type of PIM Device

Prior works only leverage
one type of PIM device with
a **certain computation and memory bandwidth**

The memory-bound FC kernels and attention kernels have **varying demands of computation and memory bandwidth**

The approach supports only one type of PIM device with a certain computation and memory bandwidth capability

Outline

Our Goal

Design a heterogeneous system that caters to the **varying parallelism levels** in real-world LLM inference with **different and dynamically changing computation and memory demands**

PAPI's Key Idea

enable **online dynamic task scheduling** on a heterogeneous architecture via online identification of kernel properties in LLM decoding

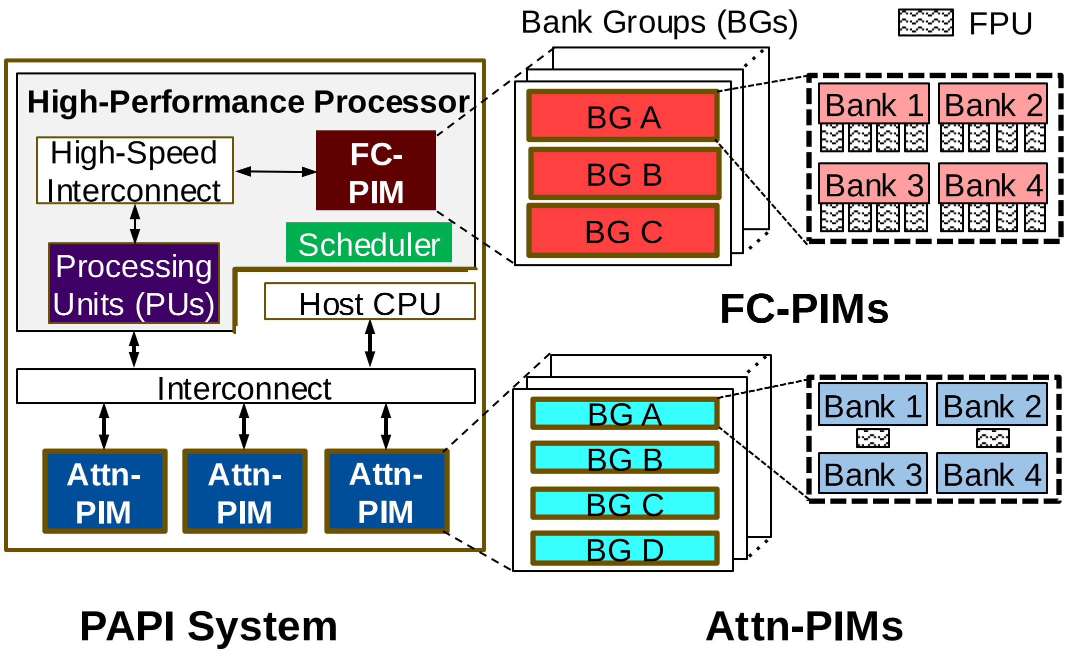
PAPI's Key Components

A new PIM-enabled computing system design

Hybrid PIM units
to cater to the different parallelism levels of the FC and attention kernels

Dynamic LLM kernel scheduling
to cater to the varying parallelism levels

PAPI's Overview



Sure <eos>
It is a good work <eos>
Have a nice day <eos>
How are you <eos>
Here is a cute dog <eos>

RLP	5	4	4	3	2	0
TLP	1	1	1	1	1	1
Reschedule	×	✓	×	✓	✓	✓
RESULT	-	**PU**	-	**PIM**	**PIM**	**PIM**

Dynamic Scheduling

Outline

1 Background

2 Observations & Motivation

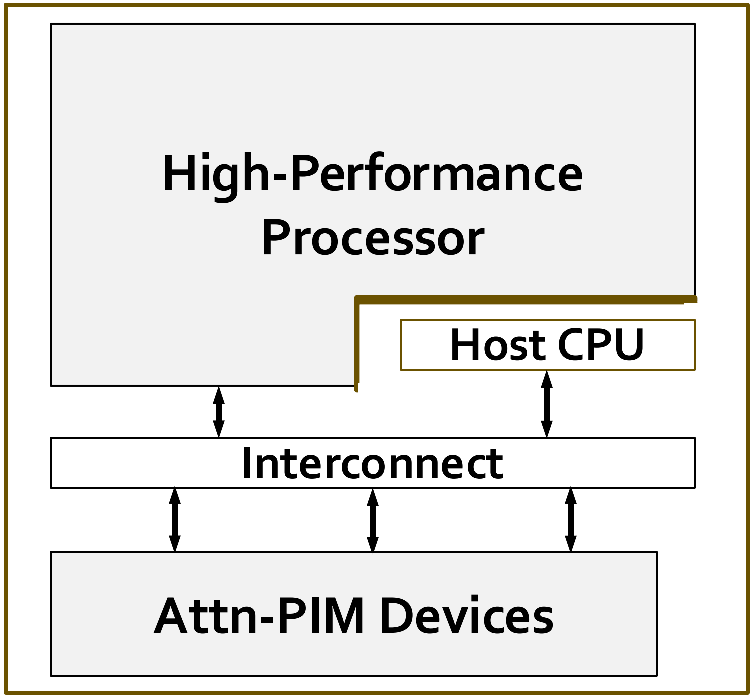
3 PAPI’s Key Idea

4 **PAPI’s Implementation**

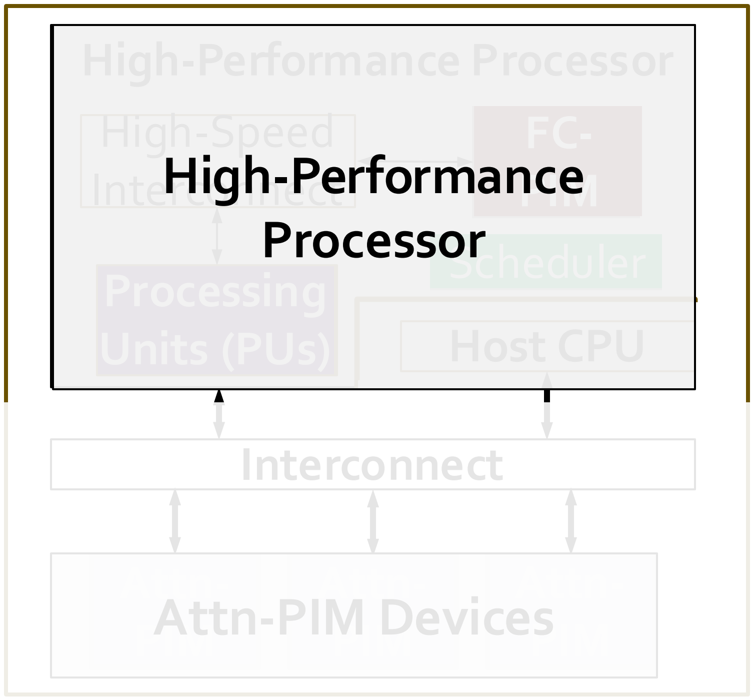
5 Evaluation

6 Conclusion

PAPI Architecture



High-Performance Processor

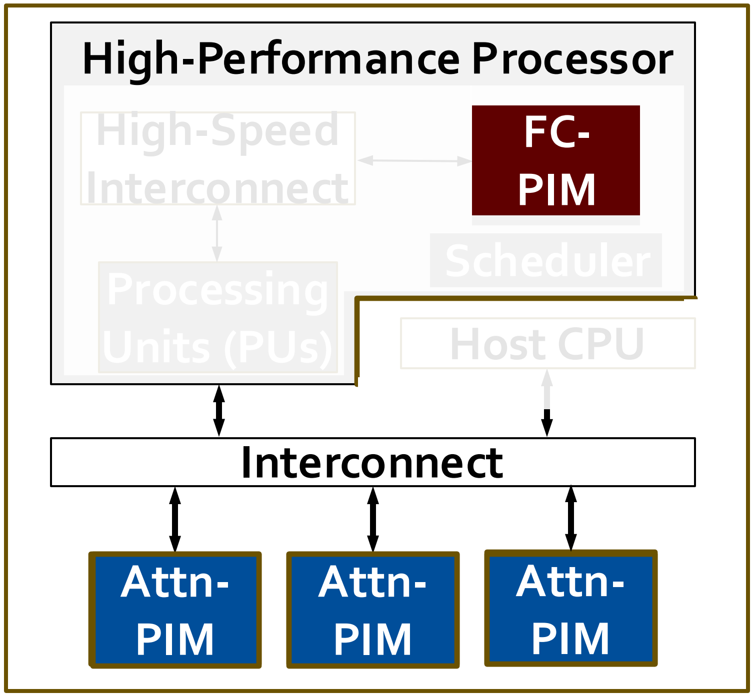


When FC kernels compute-bound:
Assign FC kernels to PUs

When FC kernels memory-bound:
Assign FC kernels to FC-PIM

FC-PIM and PUs cater to the FC kernels that
switch between memory-bound and computation-bound

Hybrid PIM Units (i)



The FC-PIM Device Placed in the High-Performance Processor

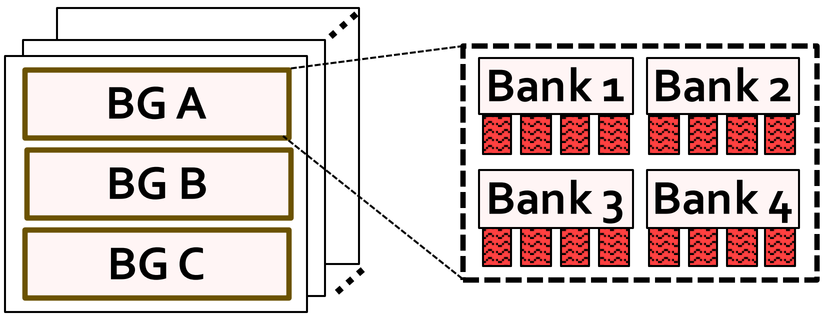
Attn-PIM Devices Disaggregated from the High-Performance Processor

Hybrid PIM units cater to memory-bound kernels with different computational demands and memory footprints

Hybrid PIM Units (ii)

Floating-Point Processing Units (FPU)

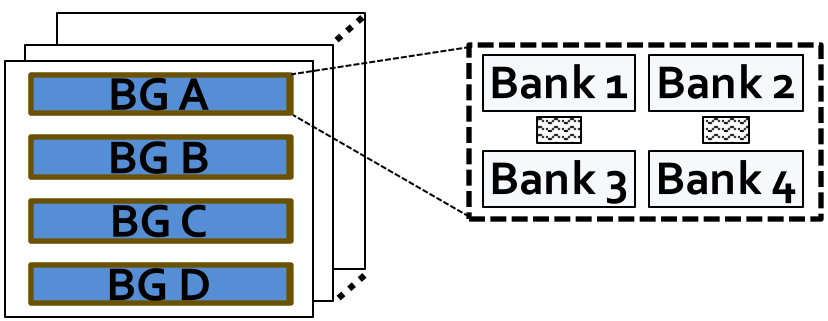
Bank Groups (BGs)



FC-PIMs

More FPUs per Bank

Higher Computation Capabilities to Cater to the FC Kernels



Attn-PIMs

More Bank Groups per Stack

Higher Memory Bandwidth for the Attention Kernels

PAPI Runtime Scheduler

Initial: memory-boundedness threshold α
(through offline iterative evaluation)

① Monitor Parallelism Levels
- RLP & TLP

↓

② Arithmetic Intensity Predictor
- Estimate arithmetic intensity of FC kernels
- Compare with memory-boundedness threshold α

↓

③ Schedule the FC Kernels
- Maps the FC kernels on FC-PIM or PU

Outline

1 Background

2 Observations & Motivation

3 PAPI’s Key Idea

4 PAPI’s Implementation

5 **Evaluation**

6 Conclusion

Evaluation Methodology

Performance and Energy Analysis:

- Simulation via AttAcc [ASPLOS'24] and Ramulator 2 [IEEE CAL]

Baselines:

- **AttAcc** [ASPLOS'24]
- **GPU+HBM-PIM** (NVIDIA A100 GPU + Samsung's HBM-PIM)
- **PIM-only** (PIM devices in AttAcc)

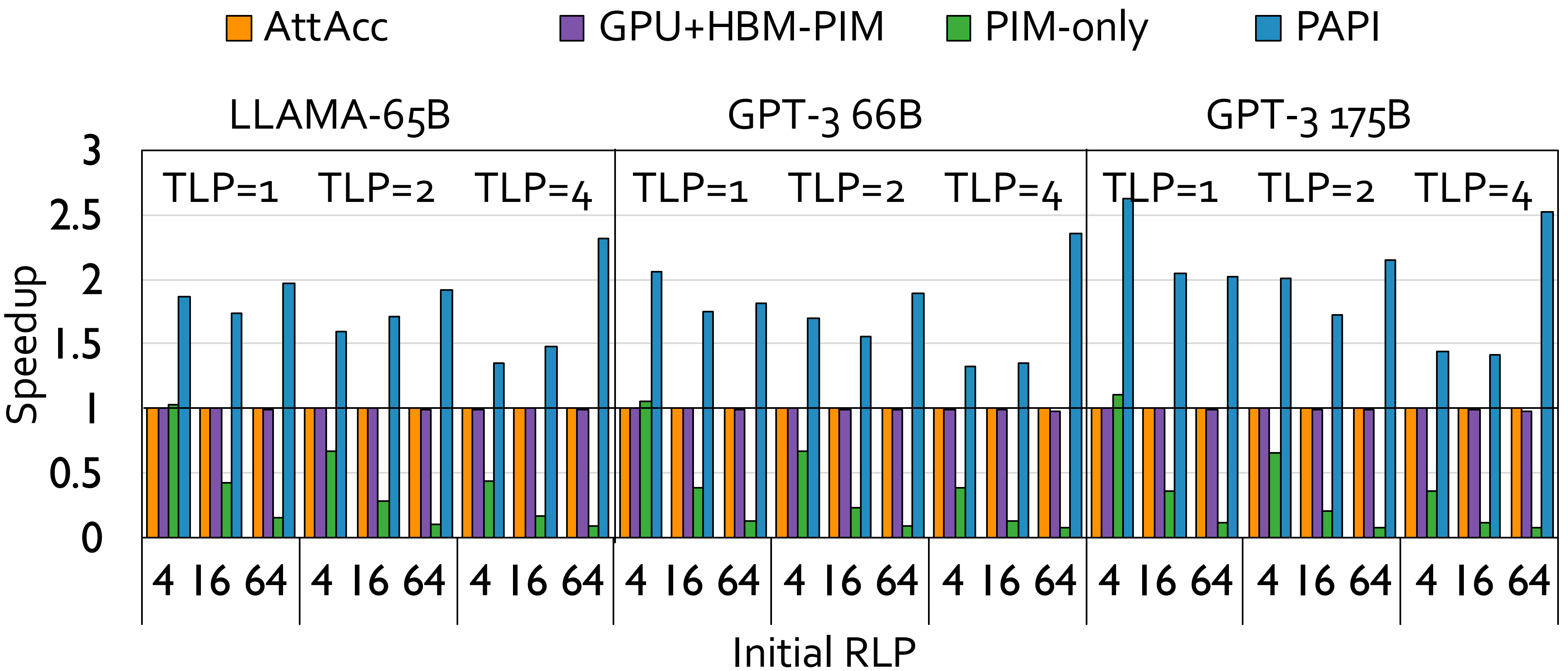
Workloads: Three transformer-based LLMs

- LLaMA-65B, GPT-3 66B, GPT-3 175B

Datasets: Dolly

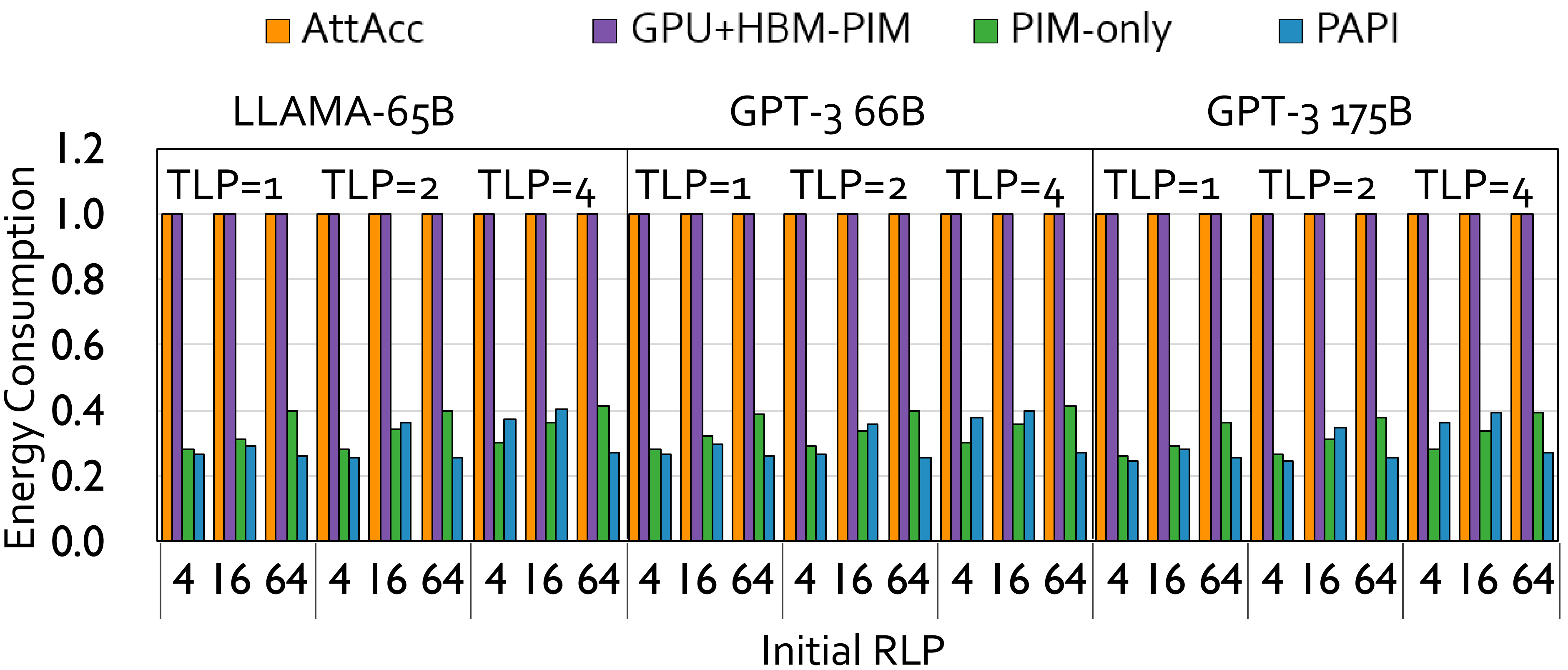
- Creative-writing tasks
- General-QA tasks

Performance Analysis



PAPI provides speedup by 1.8X, 1.9X, and 11.1X compared to the baselines

Energy Analysis



PAPI provides energy savings by 2.42X, 2.42X, and 0.15X compared to the baselines

More in the Paper

- **Details on PAPI's Heterogeneous Architecture**
- **Details on PAPI Runtime Scheduler**
- **Sensitivity to Parallelism Levels**
- **Speedup of FC-PIM**
- **PAPI's Execution Time Breakdown**

More in the Paper

PAPI: Exploiting Dynamic Parallelism in Large Language Model Decoding with a Processing-In-Memory-Enabled Computing System

Yintao He[1,2] Haiyu Mao[3,4] Christina Giannoula[5,6,4] Mohammad Sadrosadati[4]
Juan Gómez-Luna[7] Huawei Li[1,2] Xiaowei Li[1,2] Ying Wang[1] Onur Mutlu[4]
[1]SKLP, Institute of Computing Technology, CAS [2]University of Chinese Academy of Sciences [3] King's College London
[4]ETH Zürich [5]University of Toronto [6]Vector Institute [7] NVIDIA

- Sensitivity to Parallelization Levels

https://arxiv.org/pdf/2502.15470

- Performance Spee… M
- PAPI's Execution T… own

Outline

Conclusion

Key Findings

1. There are varying computation and memory bandwidth demands across different RLP & TLP configurations
2. The memory-bound kernels exhibit various computation demands depending on the kernel type
3. LLM kernels have dynamic computation demands at runtime

Key Mechanism

PAPI

Key Idea: To enable online dynamic task scheduling on a heterogeneous architecture via online identification of kernel properties in LLM decoding

Key Result: **Simultaneously improves performance and energy** efficiency of the state-of-the-art baseline

PAPI: Exploiting Dynamic Parallelism in Large Language Model Decoding with a Processing-In-Memory-Enabled Computing System

Yintao He Haiyu Mao Christina Giannoula
Mohammad Sadrosadati Juan Gómez-Luna Huawei Li
Xiaowei Li Ying Wang Onur Mutlu

ASPLOS 2025

SAFARI

The Process of Dynamic Scheduling

- Assume the memory-boundedness threshold **α=4** in this case

Output Tokens of Requests

Today	is	sunny
It	is	a
Have	a	nice
How	are	you
Here	is	a

RLP	5	5	5
TLP	1	1	1
Estimated value	5	5	5
Reschedule	✗	✗	✗
RESULT	-	-	-