D²MoE: Dual Routing and Dynamic Scheduling for Efficient On-Device MoE-based LLM Serving

MobiCom' 25

Haodong Wang, Qihua Zhou*, Zicong Hong*, Song Guo

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

深圳大学计算机与软件学院
College of Computer Science & Software Engineering, Shenzhen University

Presenter: Yunpeng Xu

2025.9.18

Author

Song Guo

Pervasive intelligence Lab

Pervasive intelligence Lab

研究方向：普适智能（Pervasive Intelligence）

研究核心聚焦于**大模型（LLMs）、边缘智能（Edge AI）、云边端协同和AI安全**的交叉优化，探索如何通过系统与算法的协同设计，实现高效、可靠、可扩展的智能服务。

- **Agentic AI Theory**
  - 高效推理（Efficient Reasoning）
  - 测试时对齐（Test-time Alignment）
  - 多智能体协作（Multi-Agent Collaboration）
- **Edge-Native AI System**
  - AI 个人计算机（Artificial Intelligence PC）
  - 云-边-端协作（Cloud-Edge-End Collaboration）
  - 国产 GPU 与 AI 芯片生态（Chinese GPU）
- **AI & Web3.0 Security**
  - AI 系统安全漏洞（AI Vulnerabilities）
  - AI 治理（AI Governance）
  - 区块链与分布式安全（Blockchain Security）
- **Pervasive AI Applications**
  - 具身智能（Embodied AI）
  - 科研赋能（AI for Sciencee）
  - AI 内容生成（AI Generated Content）

Contents

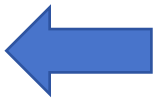
- Background & Motivation
- Challenges
- Design
- Evaluation
- Thinking

Background & Motivation

MoE-based LLM inference

The Real Bottleneck?

Computation **vs** Memory



- **Hardware.** The growth of memory capacity in edge devices **significantly lags behind** that of high-performance data centers in the cloud.
- **Model.** The **memory demands** of MoE-based LLMs have **grown rapidly** due to the expansion of model size and the need for higher performance.

Table 1: Trade-offs among different bit-width.

Bit-width	mem. (GB)	lat. (token/s)	acc. (ppl ↓)
2	3.04	50.47	20.95
3	3.80	45.91	15.10
4	4.48	43.82	14.72
5	5.10	40.15	14.63
6	5.60	37.72	14.62
8	7.24	35.34	14.55
16	13.60	23.45	14.55

How to bridge the gap?

Background & Motivation

Observation:

1. **Different bit-width** in MoE-based LLM quantization **bring different benefits** in terms of **accuracy-memory-latency.**
2. **The importance** of experts **changes dynamically** according to **different input samples.**
3. **Large bubble** between **I/O and computation** of quantized experts led to substantial **inference delays.**

Dynamically adjust the expert bit-width to align with hardware characteristics

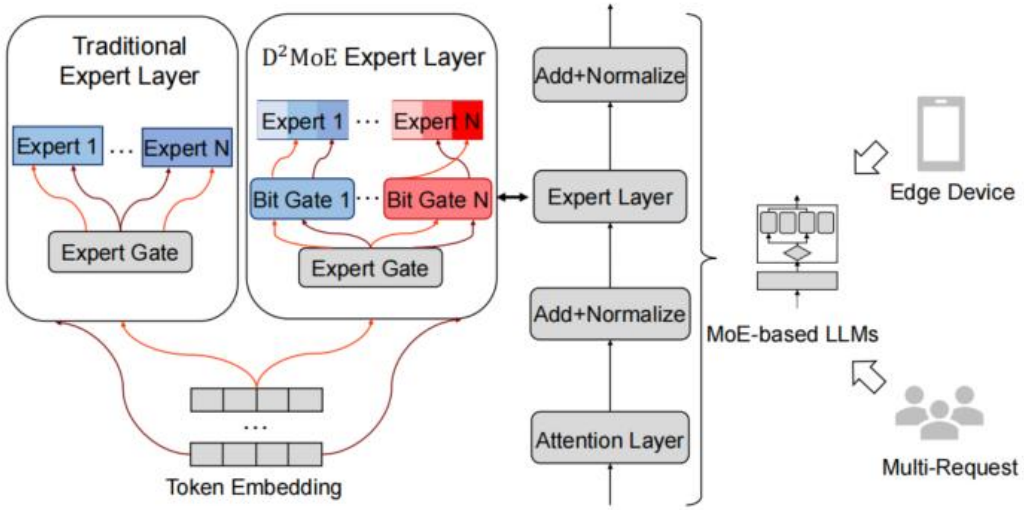


Figure 1: Traditional MoE single routing (expert ID only) vs. our D²MoE dual routing (ID and bit-wdith).

Challenges

◆ Challenge 1： **Unbalanced and inefficient** bit-width selection load

The **training** of lightweight gating network →
- Imbalanced Selection
- Irrational Allocation

◆ Challenge 2： **High memory** overhead to store multiple versions of quantized experts

The basic LLM quantization

multiple quantization **models** + **different** bit-width **versions**

→ ✖ Edge devices with **limited memory**

How To Handle?

Challenges

◆ Challenge 3: Significant runtime **overhead** on weight **dequantization** operations

During model inference:

Transfer the quantized weights into **the same data type** as the activation for matrix computation → Consume **20-70% of the entire inference latency**

◆ Challenge 4: **Inefficient** I/O–Compute **pipeline** with large parallelism **bubbles**

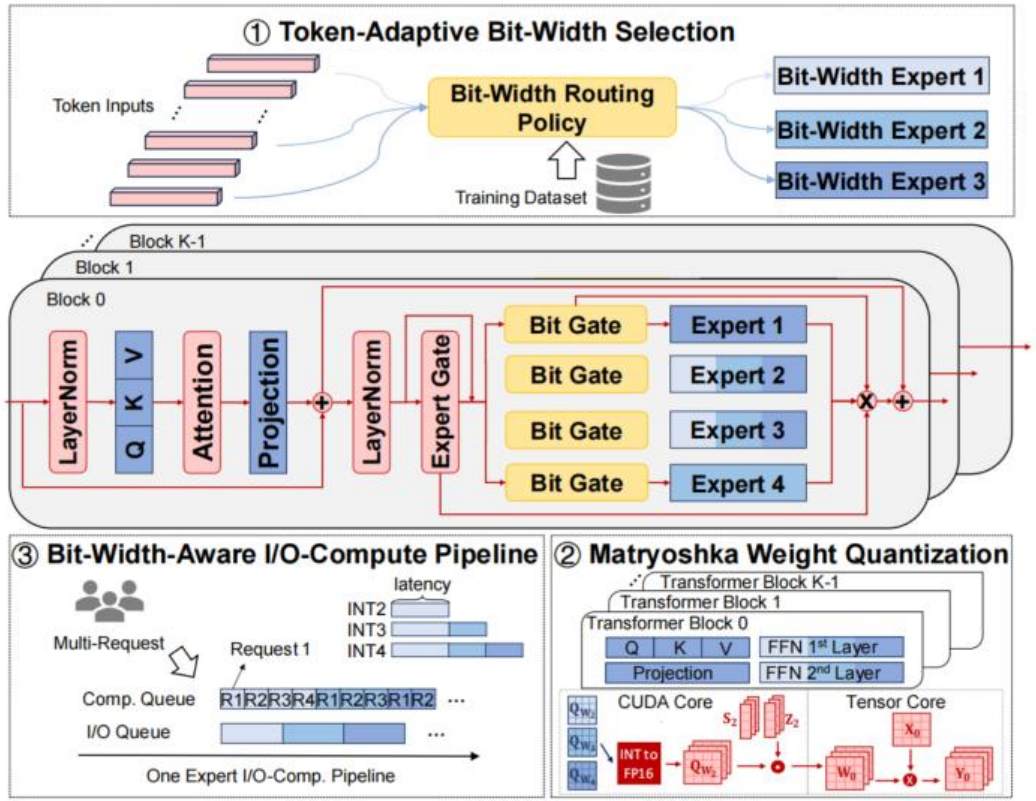
Existing methods to achieve I/O-compute parallelism:

- model placement policies
- adaptive batch scheduling

→ Quantized LLMs serving must also consider **the varying bit-width selections** for different requests

Design

System architecture of D²MoE



- **The offline preprocessing phase (On server)**

① Token-adaptive bit-width selection

-> **train** a **lightweight** gate to dynamically select the optimal bit-width expert for each token

② Matryoshka weight quantization (MWQ)

-> **compress** expert weights into a **nested, multi-bitwidth** format

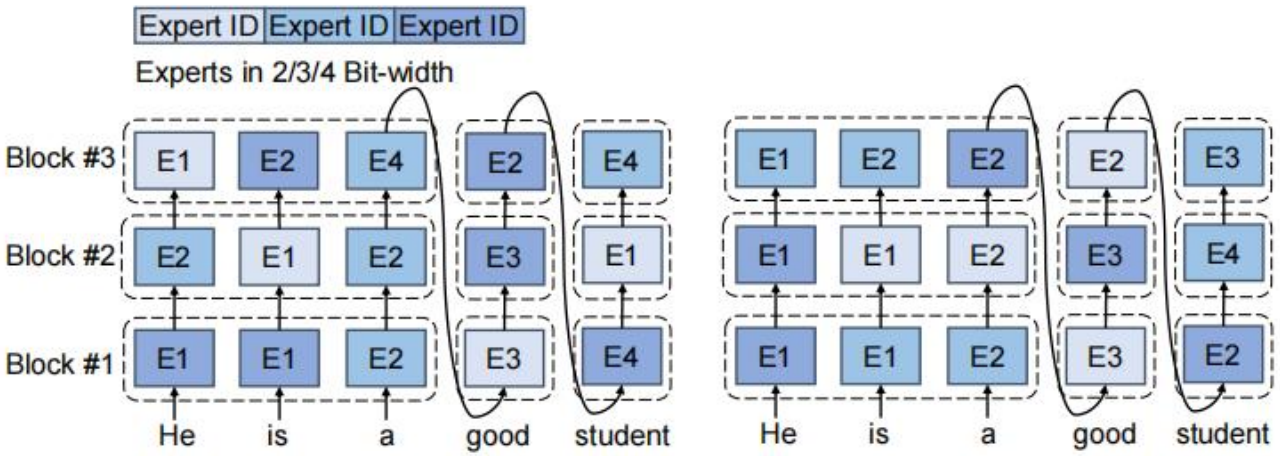
- **The online execution phase (On device)**

③ The bit-width-aware I/O-compute pipeline

-> employ the **HEBF** scheduler to overlap I/O and computation to **eliminate** the **idle bubbles**

Design

➢ Core modules 1： Token-Adaptive Bit-Width Selection



traditional method vs token-adaptve method

Two steps to deal with challenge 1 :

① Quantized Expert Capacity

Set a capacity limit **c_k** for each bit-width expert to prevent overfitting and force the router to **learn balance.**

② Dynamic Bit-Width Selection Loss

Propose a novel bit-width balancing loss to **balance the selection frequency** of different bit-width.

$$Loss = \frac{1}{T}\sum_{x\in S}\left(\underbrace{CE(p(x),q(x))}_{\text{任务精度损失}} + \frac{\alpha}{L}\sum_{l=1}^{L}\sum_{k=1}^{K}\underbrace{p_k^l(x)}_{\text{选择概率}}\underbrace{b_k}_{\text{比特成本}}\right).$$

Design

➢ Core modules 2： Matryoshka Weight Quantization

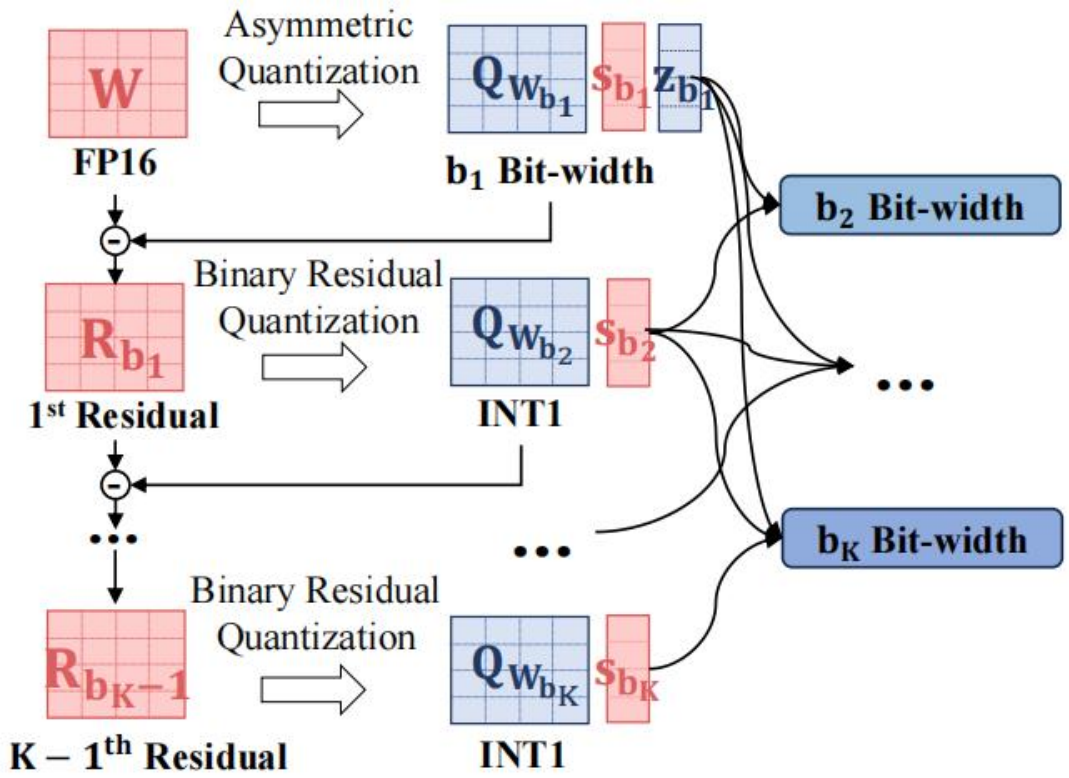


Figure 6: The workflow of MWQ.

Embed low bit-width weights within high bit-width weights to deal with challenge 2 :

① Asymmetric Quantization (to b_1)

Quantize the weights to the minimum supported bit-width (denoted as b_1 before).

② Binary Residual Quantization (b_2 to b_k)

Quantize the residual weights to increase the bit-width until the final b_k bit-width weight is obtained.

Design

Dequantization Kernel to deal with challenge 3:

tensor loading from various storage levels in the GPU

+

computation within the CUDA cores and Tensor cores

The Limited Parallelism

constrain the dequantization efficiency **on edge device**

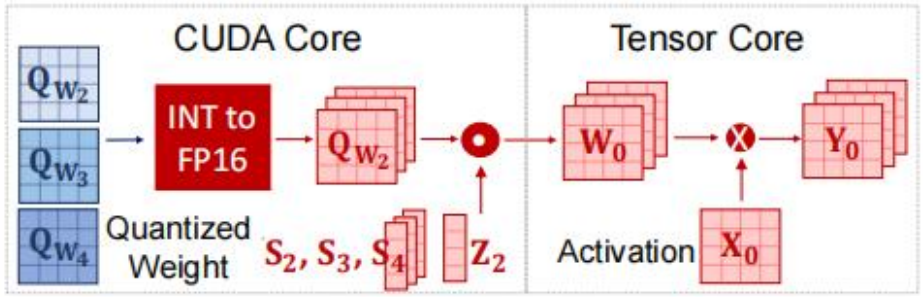


Figure 8: The dequantization overview of D^{2}MoE.

1. Loading Parallelism

① transfer quantized data from disk directly to GPU's global memory

② move activations from global memory to L2 cache

2. Computation Parallelism

① expert dequantization in the CUDA cores

② expert computation in the Tensor cores

Design

➢ Core modules 3：Bit-Width-Aware I/O-Compute Pipeline

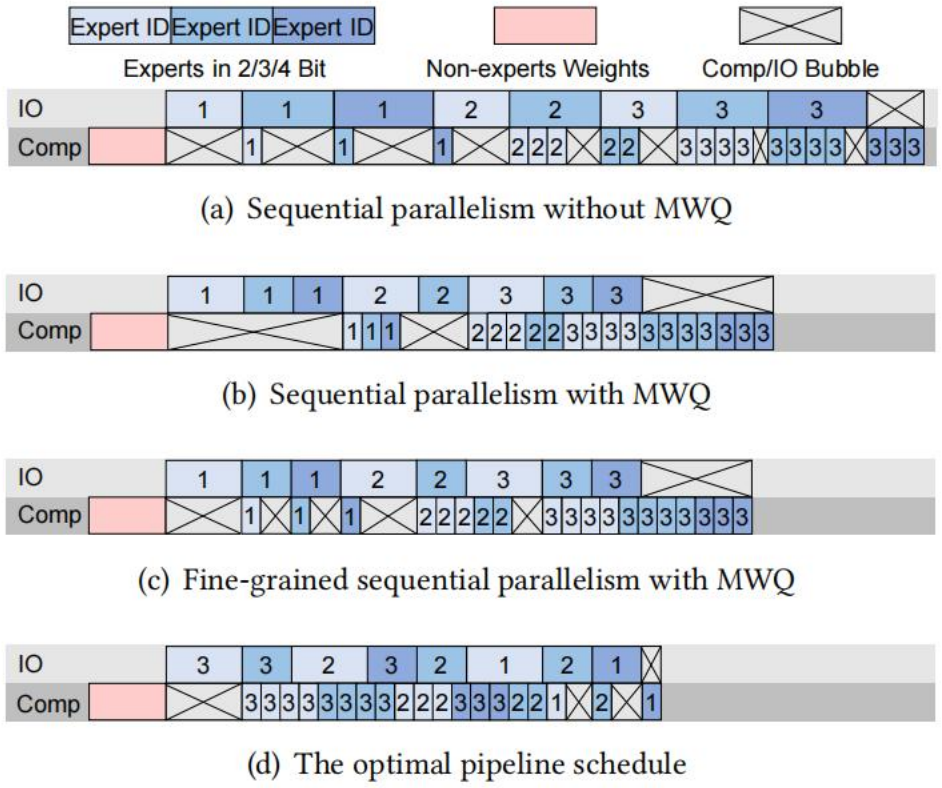


Figure 9: Comparison between different I/O-compute parallel strategies.

How to reduce the parallel bubbles?

① Low-bit-width experts can be **reused (MWQ)**

② Break the **ID-order** and dynamic scheduled by **bit-width + activation frequency (HEBF)**

↓

Maximize I/O-Compute overlap

+

Minimize idle time and end-to-end latency

Evaluation

1. Experimental Setup

- **Hardware**

Hardware	Environment 1		Environment 2	
	Device	Memory	Device	Memory
GPU	NVIDIA RTX 3060	6GB	Jetson AGX Orin	64GB
CPU	Intel Core i7-11800H	32GB	ARM Cortex-A78AE	(SoC share)
Disk	Samsung 970 EVO	1T	Samsung 970 EVO	1T
Disk Read	3.5 GB/s		3.5 GB/s	

- **Metrics**

① model accuracy → language generation performance
① model accuracy → zero-shot performance

② throughput

- **Baseline**

① Hold-in-Memory (**all** INT8 and hold in GPU memory)

② Matryoshka-Free (INT2/3/4 and load on demand)

③ Hold-in-Memory-AWQ (**all** INT4 and hold in GPU memory)

④ EdgeMoE (**static** quantization and predict to load)

⑤ MoQE-DynaIO (**uniform** bit-width and load dynamically)

- **Configuration (compared to baseline)**

① D2MoE-V1

- $b_1 = 2$ and $b_k = 4$
- expert capacity is set to {0.3, 0.4, 0.3}

② D2MoE-V2

- $b_1 = 5$ and $b_k = 8$
- the capacity of each expert is 0.25

Evaluation

2. End-to-End Results

Table 3: Perplexity accuracy (lower is better) and zero-shot accuracy (higher is better) of D^{2}MoE and the baselines in LLaMA-MoE-3.5B and Mixtral 8×7B.

Model	Method	Perplexity ↓	PiQA	Arc.e	BoolQ	HellaSwag	Winogrande
LLaMA-MoE-3.5B	Hold-in-Memory	14.55	72.32	48.76	65.56	66.34	61.77
	Matryoshke-Free	14.58	72.29	48.71	65.48	66.28	61.76
	Hold-in-Memory-AWQ	15.89	69.32	46.52	62.54	61.55	57.44
	EdgeMoE	14.78	71.46	47.36	64.43	64.32	59.52
	MoQE-DynaIO-INT4	15.66	69.34	46.52	62.58	61.53	57.40
	MoQE-DynaIO-INT8	14.55	72.32	48.74	65.58	66.28	61.71
	D^{2}MoE-V1	15.68	69.32	46.52	62.50	64.28	59.52
	D^{2}MoE-V2	14.58	72.29	48.72	65.51	66.23	61.71
Mixtral 8×7B	Hold-in-Memory	4.04	82.4	82.6	80.56	84.1	76.5
	Matryoshke-Free	4.28	80.29	80.71	78.48	82.28	74.76
	Hold-in-Memory-AWQ	4.25	80.32	81.52	78.54	83.55	75.44
	EdgeMoE	4.38	78.46	80.36	77.43	82.32	75.52
	MoQE-DynaIO-INT4	4.25	80.34	81.52	78.58	83.53	75.40
	MoQE-DynaIO-INT8	4.08	82.32	81.74	79.58	83.28	74.71
	D^{2}MoE-V1	4.28	81.32	81.52	78.05	82.88	75.52
	D^{2}MoE-V2	4.09	82.29	81.72	79.51	83.23	74.71

Model Accuracy

① language generation performance

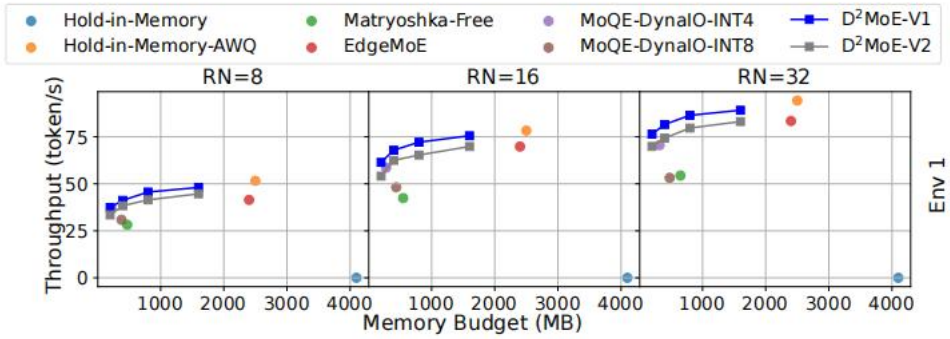
Calculate the **perplexity** to evaluate the model's **intrinsic language quality.**

② zero-shot performance

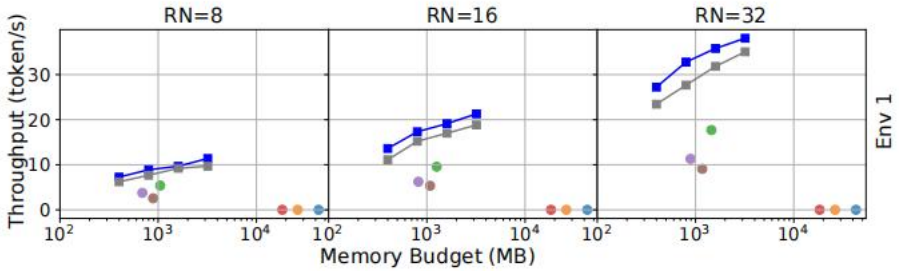
Test the model's **general capabilities** on tasks like commonsense reasoning and reading comprehension.

Evaluation

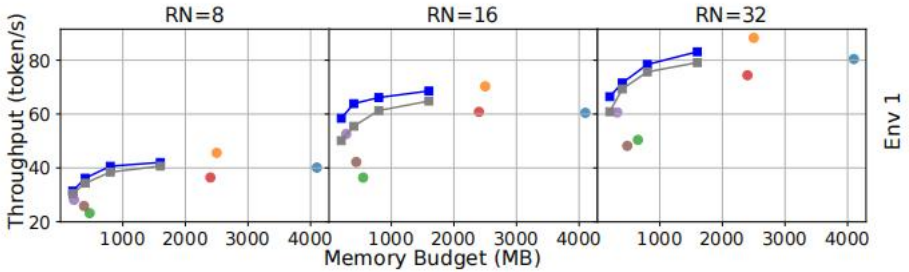
2. End-to-End Results



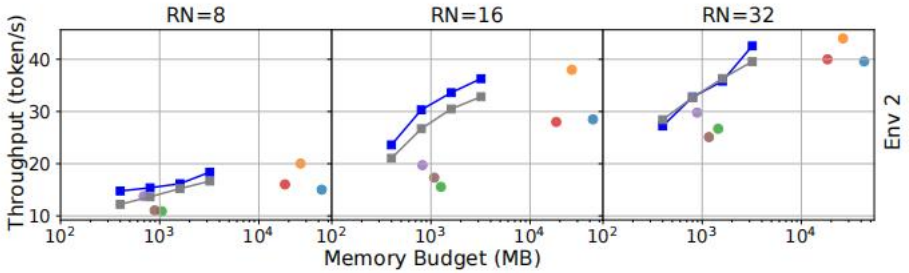
(a) Throughput of LLaMA-MoE in Environment 1.



(b) Throughput of Mixtral 8×7B in Environment 1.



(c) Throughput of LLaMA-MoE in Environment 2.

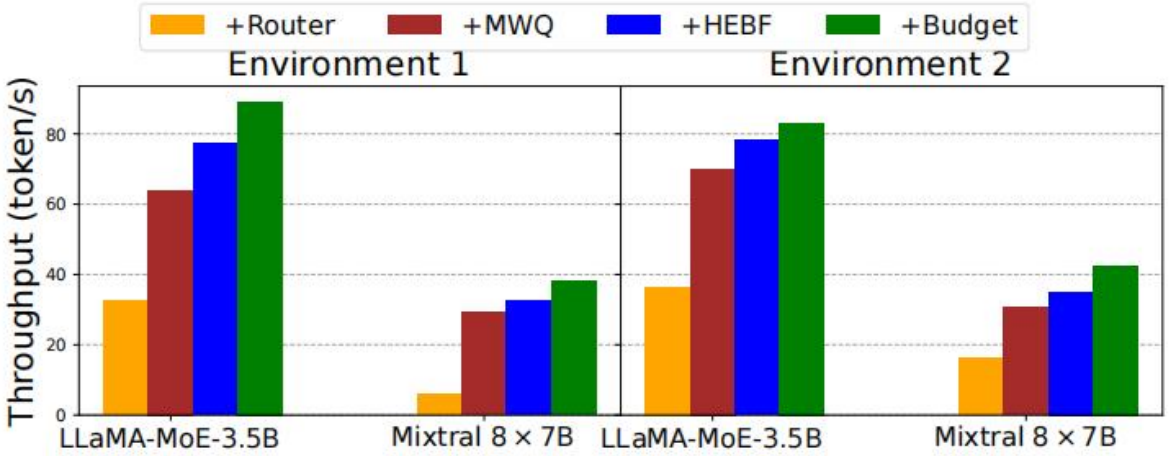


(d) Throughput of Mixtral 8×7B in Environment 2.

Throughput of D2MoE and baselines with different memory budgets

Evaluation

3. Ablation Study



+ Router: Enables **dynamic bit-width allocation** for each expert **(baseline).**

+ MWQ: Embeds lower bit-width weights into higher ones, reducing loading overhead.

→ Throughput improves **1.91×–4.95×**

+ HEBF: **Parallelizes** expert I/O and computation

→ Additional **1.11×–1.21×** improvement

+ Budget: Keeps **frequently** used experts **in GPU memory**

→ Further **1.06×–1.21×** improvement

Thinking

- **文章有什么问题，基于这篇 Paper 还能做什么优化?**

1. D²MoE 主要针对“**同步请求**”设计，未来可以探索“**异步并发请求**”的调度策略。

2. 当前系统依赖“**按需加载**”，未来可以引入“**预加载**”**机制（热度预测）**，进一步减少 I/O 等待。

3. 离线路由偏静态，**专家容量设置固定**，存在失配的可能性，未来可以引入**在线自适应容量。**

4. 在移动设备上，NPU/CPU的反量化效率需要进一步优化，需**面向NPU/CPU重新设定内核并行加载和并行计算机制。**

Thinking

- **这篇 Paper 的 Idea 能不能应用在自己的工作上面?**

1. D²MoE提出的**“动态精度”和“智能调度”**思想可以迁移到多模型推理、异构计算资源调度等领域。

2. 另外，MWQ **“嵌套量化”**的存储思想也可以迁移至资源受限的边缘设备上以减少存储开销。

- **这篇 Paper 能不能泛化?**

1. **模型架构泛化**，其他MoE变体等**稀疏模型**，甚至是**稠密模型中的FFN层**，这种**动态位宽、嵌套量化、智能调度**的思想都能有一定的性能提升。

2. **核心概念泛化**，任何**“动态决策”**系统，任何**“分层存储”或“增量更新”**系统，任何存在**生产-消费关系**的流水线系统，上述核心思想都有性能提升的空间。

Q & A

Presenter: Yunpeng Xu

2025.9.18