FlexInfer: Breaking Memory Constraint via Flexible and Efficient Offloading for On-Device LLM Inference

EuroMLSys '25, March 31, 2025, Rotterdam, Netherlands

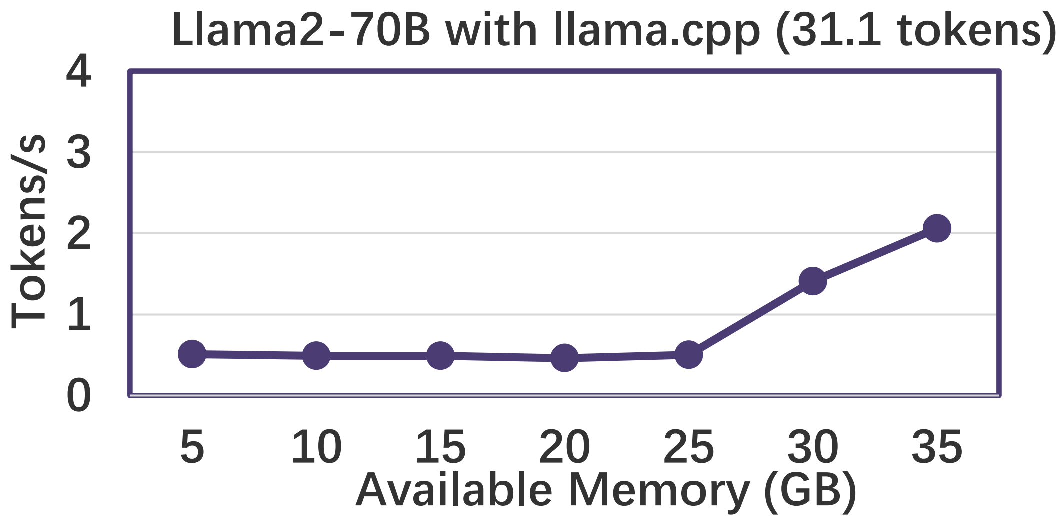
Hongchao Du, Shangyu Wu, Arina Kharlamova,
Nan Guan, Chun Jason Xue

香港城市大學
City University of Hong Kong

MOHAMED BIN ZAYED
UNIVERSITY OF
ARTIFICIAL INTELLIGENCE

Background: On-Device LLM Inference Challenges

- **LLMs demand high memory, exceeding device capacity**
  - E.g., 36.2GB for 4-bits quantized Llama2-70B
- **Cloud deployment raises privacy and customization issues**



Limitations of Current Solutions:

1. Model compression **sacrifices accuracy**
2. Offloading to storage causes **high I/O overhead**
3. Cannot adapt to **varying memory budgets**

Question: How can we achieve efficient and flexible on-device LLM inference?

Motivation: Optimizing Offloading for On-Device LLMs

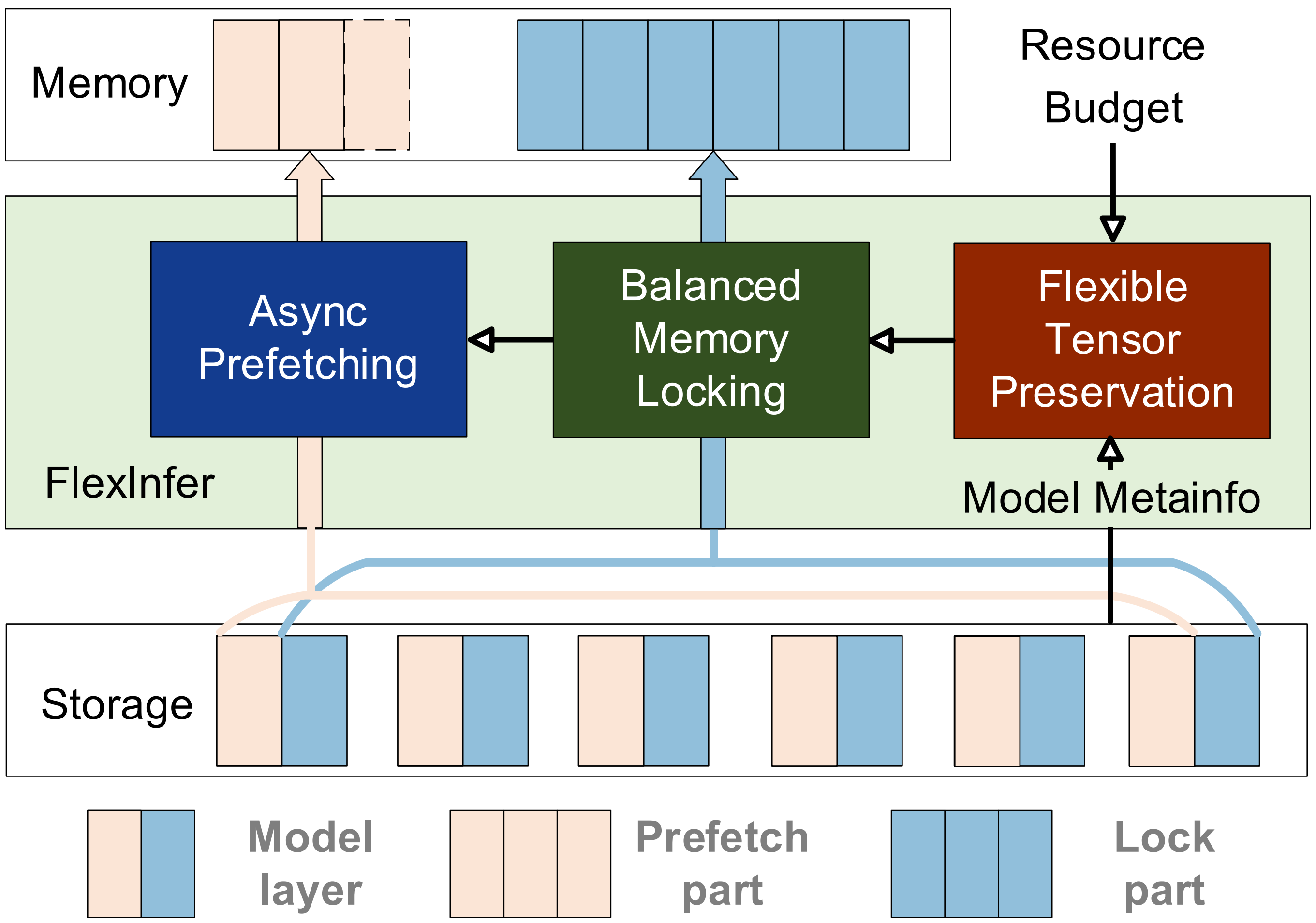
- **Current offloading suffers from high I/O overhead**
  - $T_{IO} = \frac{Size_{model}}{BW}$, $\mathrm{P}_{sync} = \frac{1}{T_{IO} + T_{CPU}}$, $\mathrm{P}_{async} = \frac{1}{\max(T_{IO}, T_{CPU})}$
  - **Key factors**: Improve I/O performance (BW) and parallelism
- **Need flexibility to adapt to varying memory budgets**
  - Maximize performance with available memory
  - **Requirement**: Adaptive fine-grained memory management method
- **FlexInfer**
  - Optimize IO performance with targeted asynchronous prefetching
  - Enhance parallelism via careful parameter partitioning
- **Targets:**
  - Performance close to the theoretical upper limit
  - Scales linearly with available memory

Outline

FlexInfer: Flexible and Efficient Offloading

- **Overview**
  - Layer-wise prefetch to optimize I/O performance
  - Tensor-level memory management for flexibility
- **Key Components**
  - 🚀 **Async Prefetching**: Overlap I/O & compute to hide latency
  - ⚖️ **Balanced Locking**: Uniform allocation to maximize parallelism
  - 🔧 **Flexible Preservation**: Retain critical tensors for any memory budget

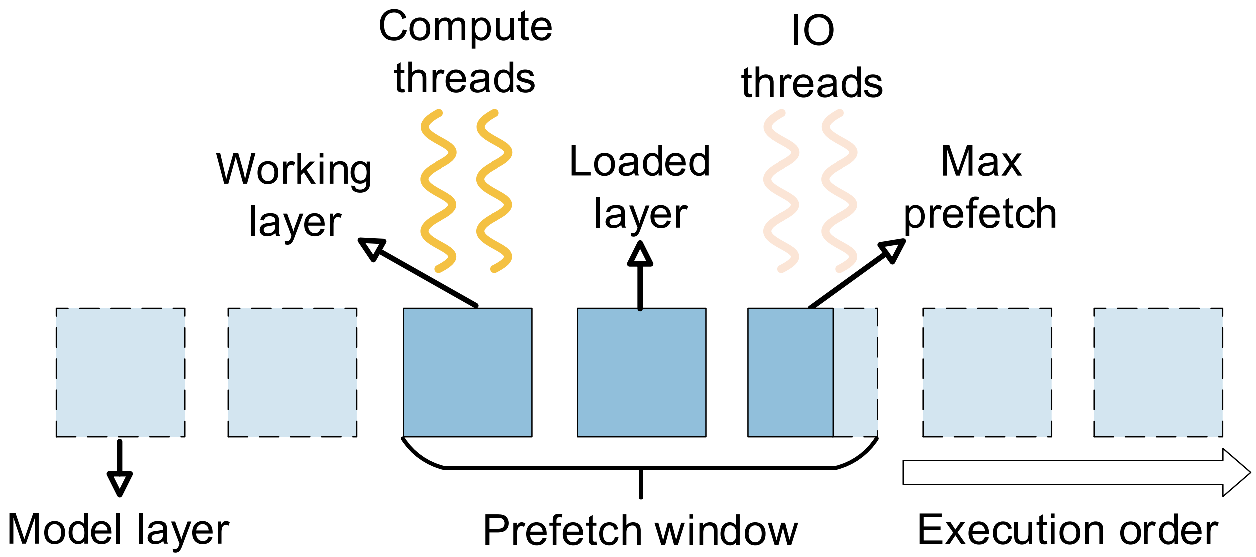
FlexInfer Workflow Overview



FlexInfer: Asynchronous Prefetching

- **Goal:** Maximize the IO efficiency of the prefetch threads
- **Mechanism**
  - 🧵 **Multi-threading**: Dedicated I/O threads prefetch next layers
  - 🗑️ **Memory Release**: Free parameters immediately after use
  - 🔒 **Atomic Sync**: Ensure correct execution order

Asynchronous Prefetching Workflow



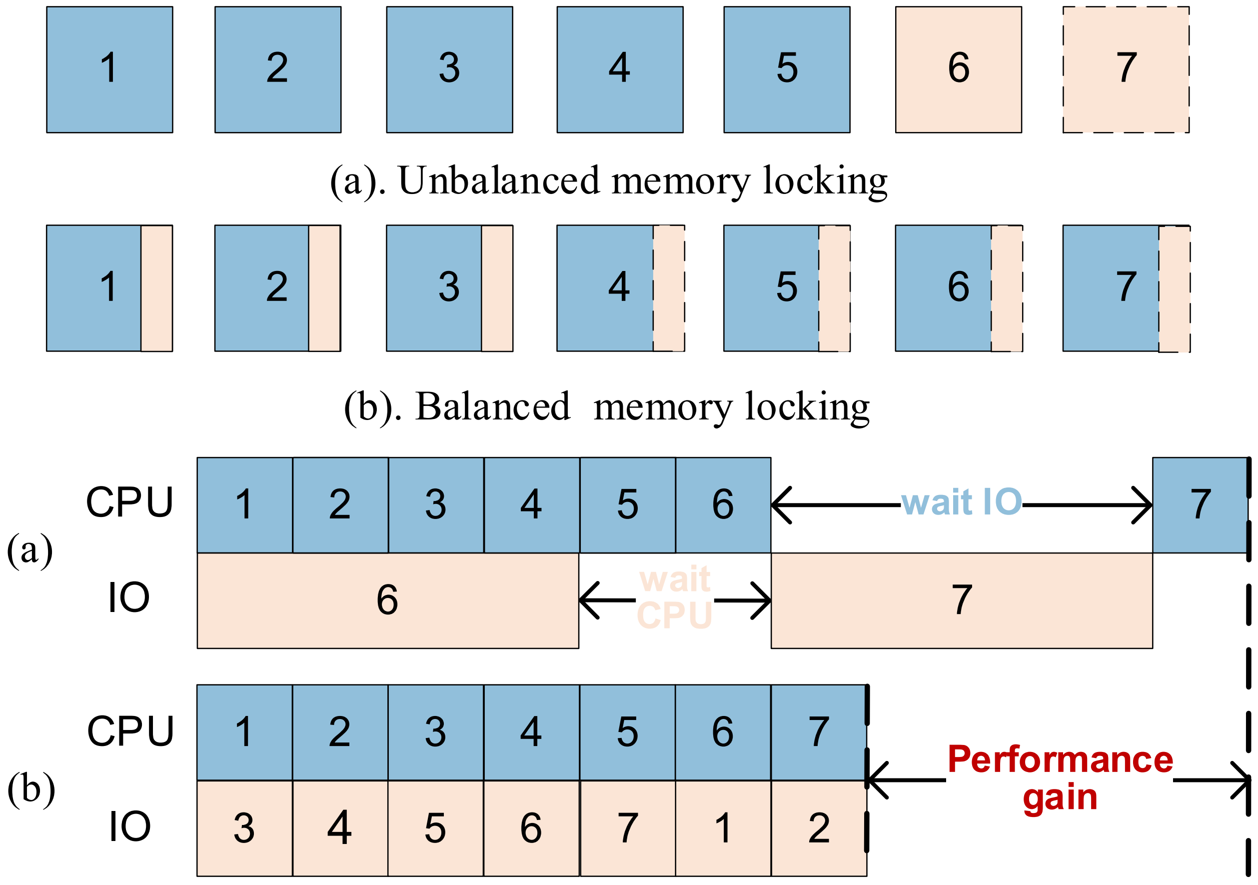
2.6~3x IO efficiency compared to mmap

34.8~59.4% faster by parallelizing computation and IO

FlexInfer: Balanced Memory Locking

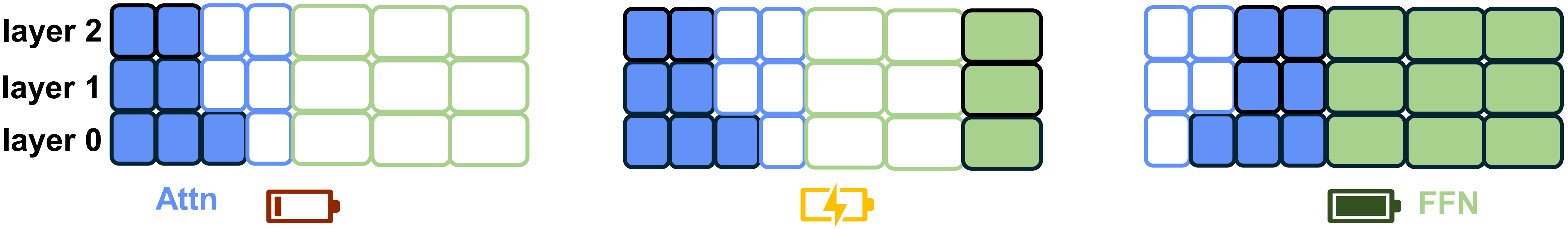
- **Mechanism:** Lock parameters using available memory
- **Key points:** Uniform allocation to maximize parallelism
- **Policy**
  1. **Unbalanced (Fails)**
     ❌ **Fixed layers cause I/O waits, reducing parallelism**
  2. **Balanced (Succeeds)**
     ✅ **Uniform splitting enables full parallelism**
     ✅ **Up to 56.8~83.3% throughput improvement over unbalanced locking**

Unbalanced vs. Balanced Locking



FlexInfer: Flexible tensor preservation

- **Observations**
  - **Attn** and **FFN** tensors impact performance differently under varying memory budgets
- **Key points:** Adopt **different lock policy** under different memory budget
  - High memory: Keep **FFN** first to keep IO uniform
  - Low memory: Keep **Attn** to reduce IO number
  - Intermediate case: Mix **FFN** and **Attn** with heuristic selection
- **Result**: Up to **21.9%** improvement compared to static policy



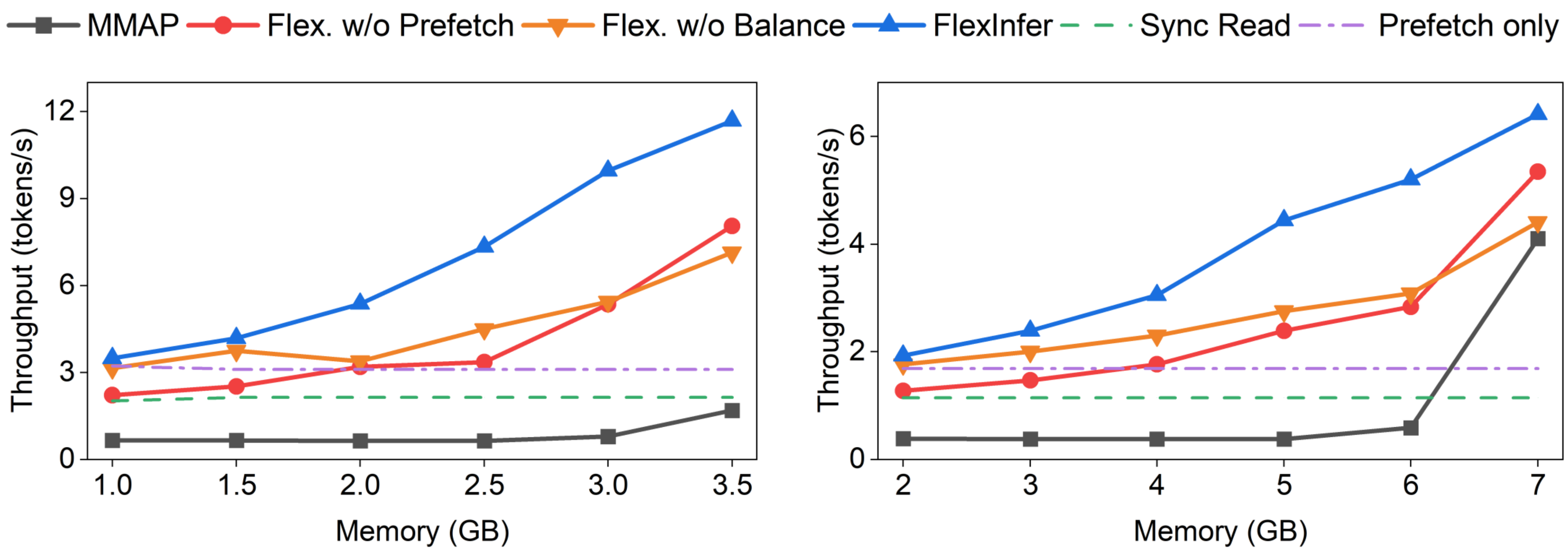
Outline

- Background and Motivation
- FlexInfer: Flexible and Efficient Offloading for LLM Inference
- Evaluation
- Conclusion

Evaluation Setup

- **Environment**
  - ⚙️ Hardware: AMD 7995WX CPU, 512GB DDR5 DRAM, 2TB Crucial T700 SSD
  - 🖥️ Software: Extended llama.cpp
    - Taskset and cgroup to simulate resource-constrained devices
- **Models & Baselines**
  - 🦙 Llama2-7B/13B/70B, Codellama-34B under 4-bit quantized
  - 🆚 MMAP, FlexInfer w/o x, Sync Read, Prefetch only
- **Metrics**
  - 🚀 Throughput (tokens/s)
- **Configurations**
  - 📥 Prefetch Window = 3 layers
  - 🛠️ CPU core number = 8 cores

Evaluation Results: Throughput

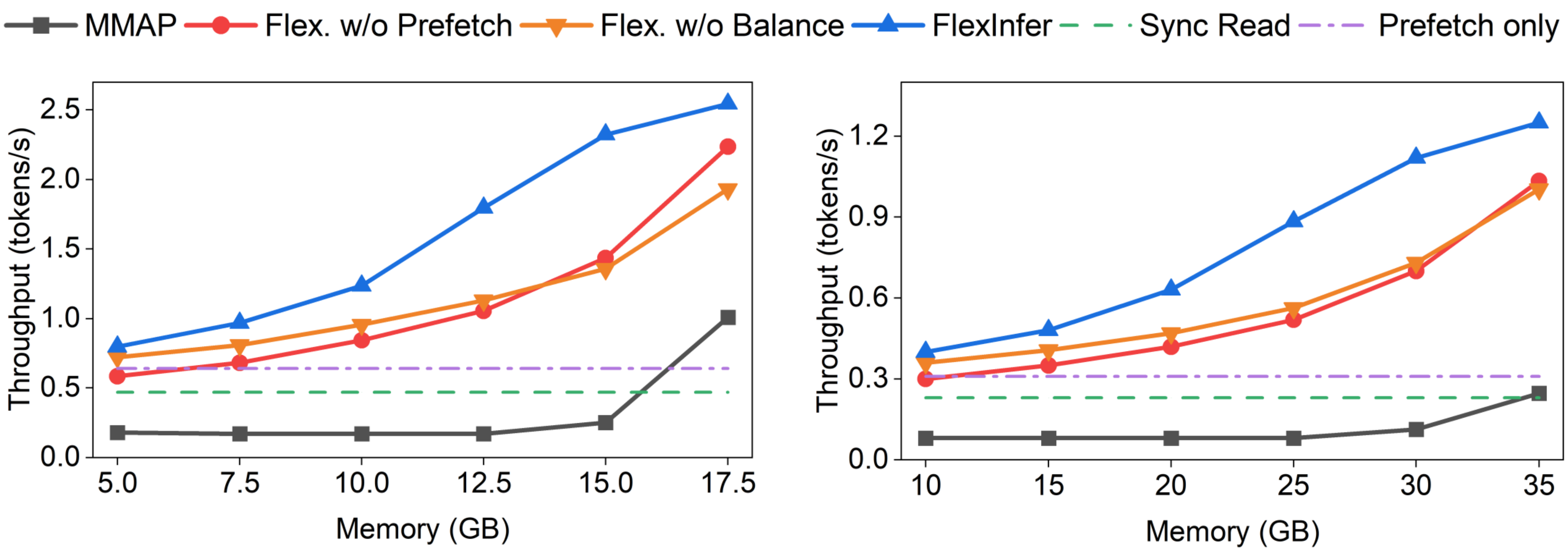


Llama2-7B (3.8GB, 12.72 tokens/s)

Llama2-13B (7.3GB, 6.74 tokens/s)

FlexInfer achieves performance improvement of 5.2-12.5x, 5-11.8x for 7B and 13B models, with nearly linear scalability

Evaluation Results: Throughput



Codellama-34B (17.9GB, 2.6 tokens/s)

Llama2-70B (36.4GB, 1.3 tokens/s)

FlexInfer achieves performance improvement of 4.2-10.6x, 5-11x for 34B and 70B models, with nearly linear scalability

Outline

- Background and Motivation
- FlexInfer: Flexible and Efficient Offloading for LLM Inference
- Evaluation
- Conclusion

Conclusion

FlexInfer Achieves:

- 🚀 2.6~3x Faster than MMAP under **Asynchronous Prefetching**
- ⚖️ Up to **56.8~83.3%** improvement via **Balanced Locking**.
- 🔧 **Any budget adaptation** with **Flexible Preservation**.

Evaluation results show that FlexInfer achieve **10.6-12.5** times inference speedup compared to mmap-based offloading

Democratizing LLMs for every edge device.

FlexInfer: Breaking Memory Constraint via Flexible and Efficient Offloading for On-Device LLM Inference

EuroMLSys '25, March 31, 2025, Rotterdam, Netherlands

Hongchao Du

Shangyu Wu Arina Kharlamova Nan Guan Chun Jason Xue

香港城市大學
City University of Hong Kong

MOHAMED BIN ZAYED
UNIVERSITY OF
ARTIFICIAL INTELLIGENCE