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

Hongchao Du*
City University of Hong Kong
Hong Kong, China

Shangyu Wu*
City University of Hong Kong
Hong Kong, China

Arina Kharlamova
MBZUAI
Abu Dhabi, United Arab Emirates

Nan Guan
City University of Hong Kong
Hong Kong, China

Chun Jason Xue
MBZUAI
Abu Dhabi, United Arab Emirates

Abstract

Large Language Models (LLMs) face challenges for on-device inference due to high memory demands. Traditional methods to reduce memory usage often compromise performance and lack adaptability. We propose FlexInfer, an optimized offloading framework for on-device inference, addressing these issues with techniques like asynchronous prefetching, balanced memory locking, and flexible tensor preservation. These strategies enhance memory efficiency and mitigate I/O bottlenecks, ensuring high performance within user-specified resource constraints. Experiments demonstrate that FlexInfer significantly improves throughput under limited resources, achieving up to 12.5 times better performance than existing methods and facilitating the deployment of large models on resource-constrained devices.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; • **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computer systems organization** → **Embedded software**; • **Software and its engineering** → **Software performance**.

Keywords: LLM, On-Device Inference, Offloading, Resource-Constrained Devices

ACM Reference Format:
Hongchao Du, Shangyu Wu, Arina Kharlamova, Nan Guan, and Chun Jason Xue. 2025. FlexInfer: Breaking Memory Constraint via Flexible and Efficient Offloading for On-Device LLM Inference. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30–April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721146.3721961

*These authors contributed equally to this work.

EuroMLSys '25, March 30–April 3, 2025, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1538-9/25/03
https://doi.org/10.1145/3721146.3721961

1 Introduction

The success of Large Language Models (LLMs) has revolutionized numerous fields, enabling breakthroughs in natural language understanding, generation, and decision-making tasks [6, 28, 36]. However, existing LLMs are typically deployed on powerful cloud-based infrastructures, which may introduce many significant issues, such as privacy concerns [31, 35], and lack of customization [25]. Deploying LLMs on edge devices is gaining a growing interest [2, 35, 44, 47], particularly in scenarios where sensitive data handling, model customization, and independent operation are crucial.

Performing on-device inference still faces significant challenges due to the substantial memory demands of LLMs [16], which often exceed the capacities of local devices [26, 40]. To reduce resource demands, existing methods propose strategies such as distilling smaller models [10, 49], applying model compression [19, 46, 50], and pruning models [2, 11, 23, 35, 42, 43, 47]. Although these approaches can improve models' memory efficiency, they inevitably impact the generality performance and still suffer in extreme resource-constrained scenarios [4, 9, 12]. Furthermore, these methods lack the flexibility to vary memory budgets or deployment constraints, requiring adjusting the hyper-parameters, such as quantization or sparsity levels, offering limited choices, and imposing overhead on adjustments.

To address memory limitations, several works leverage external storage to supplement limited device memory [2, 35, 47]. A typical way is to offload model parameters to storage devices and fetch them on demand [8, 33, 47]. However, inefficiently performing I/O operations between memory and storage would slow the inference [15, 47]. Moreover, existing offloading methods typically do not support flexible memory usage, so they also have limited adaptability to varying resource constraints. To provide flexibility for various resource-constrained environments, this paper proposes **FlexInfer**, a memory-efficient LLM inference offloading framework. FlexInfer first introduces **asynchronous prefetching** to alleviate I/O overheads and parallelize I/O operations and computations, then proposes **balanced memory locking** to uniformly retain model parameters to make full use of available memory. FlexInfer also presents **flexible tensor**

preservation to determine what model parameters should be offloaded and retained based on user-specified resource budgets. Those techniques perform precise memory management and enhance IO efficiency. We conducted extensive experiments to show that the proposed FlexInfer can achieve 10.6-12.5 times inference speedup compared to existing offloading techniques across various memory-limited scenarios.

In summary, this paper makes the following contributions:

- We propose FlexInfer, a novel framework that optimizes offloading-based on-device inference for LLMs through asynchronous prefetching, balanced memory locking, and flexible tensor preservation.
- We develop precise memory management strategies to minimize I/O bottlenecks and maximize memory efficiency, enabling the deployment of large models on resource-limited environments.
- Extensive experiments demonstrate that FlexInfer significantly outperforms existing methods with high throughput under varying user-specified budgets.

2 Background and Motivations

2.1 LLM Inference

Existing LLMs basically adopt the transformer-based architecture [38], which consists of multiple transformer blocks. Each transformer block contains a self-attention module and a feedforward module. Given a token sequence $X = [x_1, x_2, \ldots, x_n]$, where each x_i is a d-dimensional vector, the self-attention module computes three internal states, i.e., $Q = XW_Q^T$, $K = XW_K^T$, and $V = XW_V^T$, then the outputs of the attention modules can be computed as,

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{1}$$

where d_k is the dimension of the keys. For the feedforward module, it computes the output with two linear transformations and a nonlinear activation function.

$$FFN(h) = ACT(hW_{up} + b_{up})W_{down} + b_{down}, \tag{2}$$

where h is the outputs of the attention modules, $ACT(\cdot)$ is the activation function such as SwiGLU in Llama-series models [7, 36, 37]. $W_Q, W_K, W_V, W_{up}, b_{up}, W_{down}, b_{down}$ are learnable parameters, and all of these parameters are required when generating each token.

2.2 Deployment on Memory-Constrained Devices

Mobile devices and edge computing platforms are crucial to enable real-time, low-latency interactions with LLMs. However, these devices are typically constrained in terms of memory and processing power compared to cloud-based systems or high-performance servers. The massive number of parameters in modern LLMs often exceeds the memory capabilities of these devices. For example, state-of-the-art models such as Llama series models [7, 36, 37] have billions of parameters, resulting in memory footprints that can easily surpass the available memory of most mobile devices.

Table 1. Inference throughput (tokens/second) on memory-constrained scenarios. The size of 'Llama2-70B' is about 36.2 GB, and the full-memory throughput is 31.14 tokens/s.

Ava Mem	5	10	15	20	25	30	35
Llama2-70B	0.51	0.49	0.49	0.46	0.50	1.41	2.06

Several strategies have been proposed to reduce memory and computational demands, e.g., adopting smaller models [10, 49], model compression [19, 46, 50], and model pruning [2, 23, 35, 42, 43, 47]. While these approaches reduce resource requirements, they share two critical limitations: **Lack of Flexibility**: These methods cannot directly adapt to varying memory budgets or constraints. The required memory size becomes fixed once parameters such as model size, quantization levels, or sparsity thresholds are determined. **Limited Scalability**: Despite these optimizations, large-scale models that exceed available memory remain unsupported. To support models exceeding available memory, offloading-based methods reduce memory usage by moving parts of data stored in the memory to external or slower memory. However, this will bring non-negligible IO overhead, and careful design is required to reduce performance loss.

2.3 Motivations

To show the impact of deploying memory offloading in LLM inference under memory-constrained scenarios, we conducted preliminary experiments with the state-of-the-art inference engine llama.cpp [8] and used cgroup [20] to limit the available memory. The default serving method in llama.cpp is mmap [21], which loads the corresponding data from the storage with page faults if it is not in the memory when accessed. Table 1 presents the inference throughput of the 4-bit quantized llama-2-70b chat model [37] under various memory-constrained scenarios. The results indicate a substantial decrease in inference performance when memory is not sufficient. This is because almost every access to the model weight triggers IO operations. Providing more memory improves the inference performance to some extent, but it is still far from the performance of full memory. In summary, the offload-based on-device inference must address the following challenges: high-cost IO operations, inability to utilize available memory fully, and lack of flexibility and scalability for different memory sizes.

3 FlexInfer

This section introduces the design of FlexInfer, an offloading-based framework for efficient on-device inference of LLMs

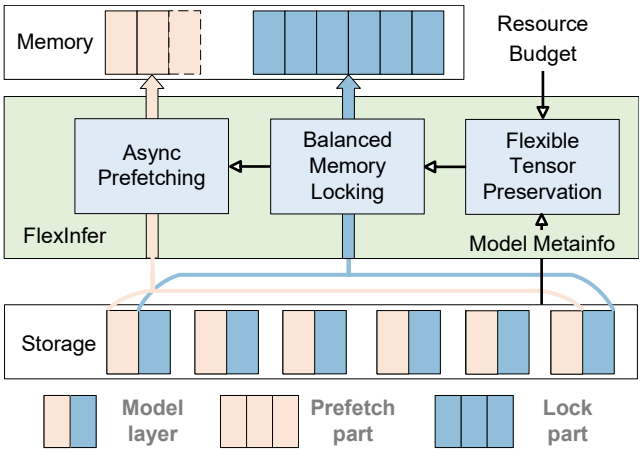


Figure 1. FlexInfer architecture.

under resource constraints. We first present the system architecture (§3.1), followed by the asynchronous prefetching mechanism that optimizes the overlap of I/O and computation (§3.2). We then detail the balanced memory locking strategy for efficient memory management (§3.3) and the flexible tensor preservation technique for intelligent parameter selection (§3.4). Finally, section 3.5 discusses implementation considerations.

3.1 Architecture Overview

As illustrated in Figure 1, FlexInfer consists of three key components, which operate collaboratively during inference: the flexible tensor preservation optimizer first determines the parameter preservation plan based on the available budget and model metadata information, and the memory locking manager then divides the model into two parts: one is loaded and fixed in memory, while the other is loaded on demand through prefetching, handled by the asynchronous prefetch module. The detailed design and implementation of each component are presented in the following sections.

3.2 Asynchronous Prefetching: Reducing I/O Overhead

When sufficient memory is available, all model parameters can reside in memory, eliminating the need for storage I/O operations. If memory constraints prevent full model loading, offloading-based methods leverage storage devices as an extension of memory, dynamically loading required parameters. In this scenario, inference performance is significantly influenced by I/O overhead, the model throughput (tokens/s) with synchronous offloading can be expressed as[1]:

$$T_{sync} = \frac{1}{Per\_token\_CPU\_latency + \frac{IO\_size}{IO\_bandwidth}} \quad (3)$$

[1]This paper focuses on CPU inference since resource-constrained devices usually don't have powerful GPUs.

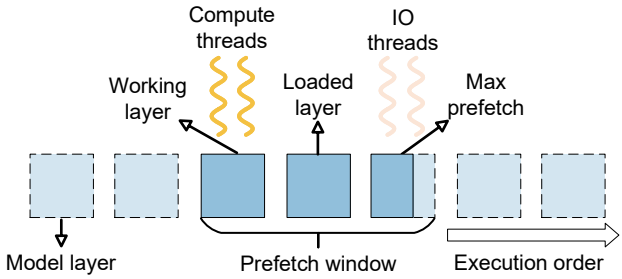


Figure 2. Asynchronous prefetching.

One common optimization is to parallelize I/O and computation operations, leading to an improved theoretical performance model:

$$T_{async} = \frac{1}{\max(Per\_token\_CPU\_latency, \frac{IO\_size}{IO\_bandwidth})} \quad (4)$$

This formulation reveals that optimal performance in offloading-based methods depends on two key factors: maximizing the parallelization between I/O and computation threads and fully utilizing available storage bandwidth.

FlexInfer employs a tensor-based multi-threaded prefetching strategy to achieve efficient parallelization and high bandwidth. We maintain input and output embedding layers in memory, focusing our strategy on handling identical-sized decoding layers. The computation threads process a layer only after the I/O threads load their parameters, with synchronization managed through atomic operations on shared variables. Multiple IO threads or computing threads will collaborate to process a particular layer and move to the next layer together, with each I/O thread responsible for loading a single tensor (e.g., W_Q, W_K, W_V.). This multi-threaded IO operation at the tensor-level granularity helps avoid inefficient small-size data transfers, optimizing bandwidth utilization.

A key observation in large model inference is that each parameter is accessed precisely once during token generation, eliminating any potential for parameter locality optimization. Our offloading strategy leverages this characteristic by immediately releasing the memory after parameter usage. The total memory footprint is thus determined by the size of the prefetch window, as shown in Figure 2. Consequently, our offloading method achieves a memory reduction ratio of approximately $\frac{k}{n}$ compared to the original model size, where n represents the number of model layers and k is the prefetch window size.

3.3 Balanced Memory Locking: Maximizing Memory Efficiency

The offloading-based method effectively reduces memory usage but is constrained by IO overhead, which limits performance. Moreover, increasing the available memory does not improve performance, as prefetching alone cannot reduce the required IO during each inference cycle. To address

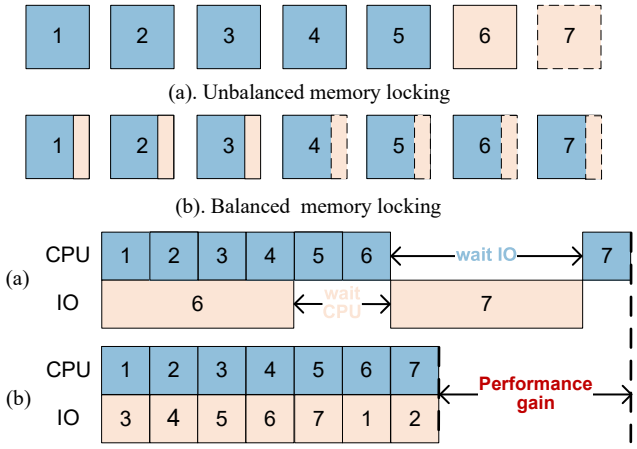


Figure 3. Balanced Memory Locking.

this challenge, FlexInfer introduces an adaptive memory-locking strategy. This strategy leverages excess memory to retain specific parameters in memory, reducing the *IO_size* in formula 4 and bringing better performance. However, a critical aspect of this approach is selecting the appropriate parameters to retain in memory for optimal results.

A naive approach might involve retaining several layers of the model in memory, thereby directly removing the IO needed for those layers. However, such an uneven memory-locking strategy introduces variability in processing speed across layers, causing the computation and IO threads to wait on each other. For example, as illustrated in Figure 3(a), a layer-based memory-locking method retains the first five layers entirely in memory while using the remaining memory to store one of the last two layers. This imbalance leads to IO thread delays until the compute thread releases memory and subsequent compute thread delays while waiting for IO to complete. This hinders the system's ability to achieve complete parallelism between IO and computation, ultimately affecting overall performance.

To address this issue, the balanced memory-locking strategy divides each layer into two parts: one fixed in memory and the other dynamically prefetched, as shown in Figure 3(b). By distributing memory usage uniformly, the IO workload for each layer remains stable throughout the inference process, enabling consistent and efficient parallelization of IO and computation. By maintaining a balanced locking approach, FlexInfer achieves significantly better performance, minimizing unnecessary delays and maximizing resource utilization.

3.4 Flexible Tensor Preservation: Heuristic Parameter Management

When the tensor sizes within each model layer are the same size, balanced memory locking can evenly distribute available memory across all layers. However, varying tensor sizes in LLMs affect performance depending on which parameters

Algorithm 1 Flexible Tensor Preservation Algorithm.

Input: Attention tensor size $size_{atte}$, FFN tensor size $size_{FFN}$, Layer number N, Memory budget $size_{mem}$,
Output: Tensor preservation plan P

```
if size_mem >= size_FFN * N * 3 + size_attn * N * 2 then
    Set all FFN tensors for all layers
else
    if size_mem >= size_FFN * N * 2 then
        Set two FFN tensor for all layers
    else
        if size_mem >= size_FFN * N then
            Set one FFN tensor for all layers
        end if
    end if
end if
Set as much as possible attention tensors one by one
return P
```

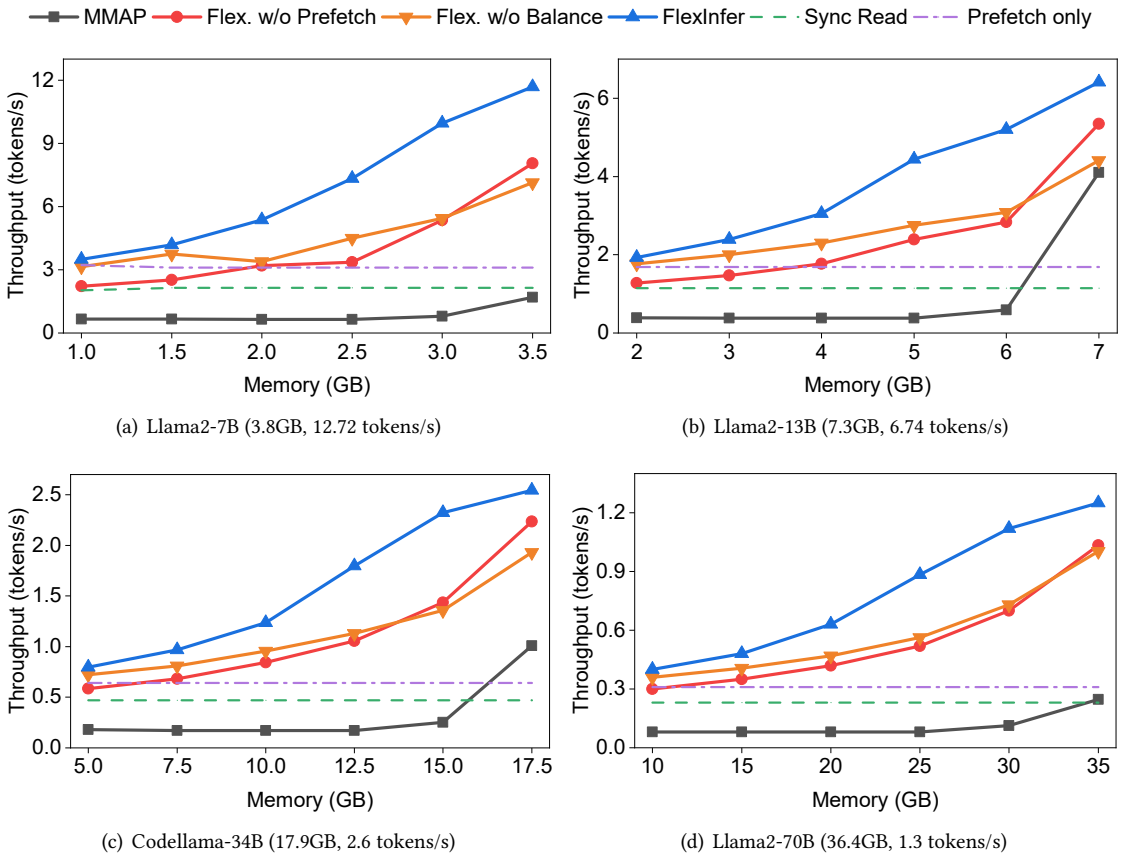
are kept in memory. The transformer architecture provides a consistent tensor structure across LLMs. Most parameters are associated with attention tensors (W_Q, W_K, W_V) and FFN tensors ($W_{up}, b_{up}, W_{down}, b_{down}$), typically with an approximate 1:3 size ratio between one attention tensor and one FFN tensor. Attention and MLP tensors have their own advantages and disadvantages under different available memory sizes. For example, when available memory is small, prioritizing attention parameters can save as many tensors as possible in memory, reduce the number of IO operations, and implement larger-size IO through FFN tensors. When memory is considerable, saving all FFN tensors can minimize memory fragmentation and the difference in residual size between layers, keeping each layer's IO overhead uniform.

Leveraging this predictable tensors structure of LLMs, we developed a heuristic algorithm, outlined in Algorithm 1. The core idea is to select parameters to retain in memory based on the available memory size, which is shown as follows:

- When memory is sufficient: If the available memory can accommodate all FFN parameters and half attention tensors, FFN tensors are prioritized and fully retained.
- When memory is limited: Attention tensors are prioritized if memory is insufficient to hold one FFN parameter for all layers.
- Intermediate cases: When memory falls between these two extremes, FFN parameters are selected incrementally until the remaining memory cannot hold one FFN tensor for all layers. At that point, as many attention parameters as possible are retained.[2]

This heuristic ensures that the remaining parameters across layers and any unused memory fragments do not exceed

[2]For models that apply GQA, we prefer smaller W_k, W_v compared to W_q, W_o.



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

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

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

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

Figure 4. Evaluation result.

the size of a single attention tensor. When the remaining memory cannot hold an extra tensor for all layers, we prioritize the attention tensor to reduce the differences across layers and prioritize the large-size IO of FFN. This design balances memory utilization and IO efficiency, optimizing the inference process with a simple strategy.

3.5 Implementation

The FlexInfer framework was implemented with 828 lines of C/C++ code, extending llama.cpp [8] to incorporate asynchronous prefetching, balanced memory locking, and flexible tensor preservation. Additional parameters were introduced to control the available memory and configure the number of threads. The default size of the prefetch window is set to 3, ensuring efficient memory management and minimal latency during inference. We use direct IO to bypass the page cache for IO threads.

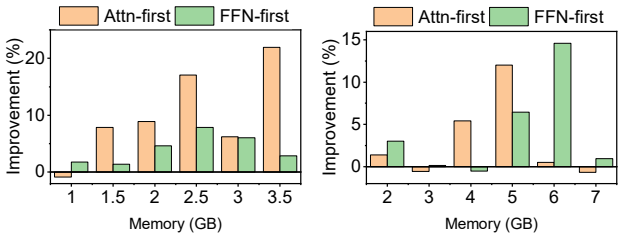
4 Evaluation

4.1 Experimental Setup

To show the results under different available memory conditions, we tested the performance of FlexInfer on an Ubuntu server with 512GB memory and AMD 7995WX CPU. We used cgroup [20] and teskset [22] to limit the available memory and CPU cores to simulate resource-constrained devices. In addition to the mmap baseline (*MMAP*), we tested FlexInfer without prefetching (*Flex. w/o Prefetch*) and FlexInfer without balanced memory locking (*Flex. w/o Balance*) under different available memory conditions to show the effect of prefetching and balanced locking. *Flex. w/o Prefetch* loads all parameters synchronously into memory before use and *Flex. w/o Balance* locks the model parameters by layer order. At the same time, to show the impact of adaptive memory locking, we also tested the results of reading parameters synchronously separately (*Sync Read*) and prefetching separately (*Prefetch only*).

4.2 Inference Throughput

The decoding performance of Llama2 series models [37], including llama2-7B, llama2-13B, Codellama-34B, and llama2-70B, are shown in Figure 4. The total memory size required to run each model and the performance when sufficient memory is available are shown in the title of each subfigure. Experimental results show that although mmap can run with very little memory, it can only achieve very limited inference performance, with only 0.08-0.67 tokens/s for different



(a) Llama2-7B (b) Llama2-13B

Figure 5. Ablation study for flexible tensor preservation.

models. When the available memory increases, the performance slightly improves. This is because mmap causes all parameters to be loaded into memory through inefficient synchronous IO, and the parameters loaded into memory are swapped out of memory before the next use, resulting in very limited scalability brought by more memory. In contrast, FlexInfer effectively improves the inference performance under different memory conditions, achieving performance improvements of 5.2-12.5x, 5-11.8x, 4.2-10.6x, and 5-11x than mmap under models of different sizes.

4.3 Ablation Studies

By replacing mmap with multiple threads large reads (*Sync Read*), FlexInfer can improve performance by 2.6-3x when memory is limited, proving that IO is the main bottleneck for large model inference under memory-constrained conditions. By introducing prefetching, FlexInfer further improves performance by 34.8-59.4% by parallelizing computation and IO. As available memory increases, the performance improvement achieved by prefetching can reach up to 69.9-118.8%. Balanced locking has a limited improvement over unbalanced memory locking when memory is low, ranging from 9.2-11.1% at minimum memory settings. This is because only a small number of parameters are locked in memory, so the difference between different strategies is limited. With more memory, balanced locking can improve by up to 56.8-83.3%.

For the ablation experiment of the parameter preservation algorithm, we compared two simple strategies, *Attn-first* and *FFN-first*, which prioritize the attention parameters and FFN parameters to be retained in memory. Figure 5 shows the performance improvement of our method over the simple strategy at different memory sizes. We only provide results for the 7B and 13B models since the 34B and 70B models employ GQA [1], which makes our strategy produce the same optimal results as *Attn-first* in most cases. Experimental results show that FlexInfer can achieve up to 21.9% and 7.8% performance improvement on 7B and 13B models, respectively, compared with Attn-first, and 12% and 14.6% performance improvement compared with FFN-first.

5 Related Work

5.1 LLM Inference Serving Engines

With the rapid development of LLM, many model-serving systems [27, 30, 41, 48, 51] and model-serving optimization [3, 5, 13, 14, 17, 24, 29, 32, 34, 39] have been proposed. However, most of them are optimized for server environments with powerful GPUs and batch-based workloads. Among them, Hugging Face Accelerate [13] and DeepSpeed Zero [3] support offloading model parameters to CPU memory or SSD. However, they still rely on GPU and are unsuitable for running on edge devices. FlexGen [34] studies swapping model parameters between GPU, CPU, and SSD to alleviate memory requirements but still targets batch-based workloads and requires at least one GPU. Inspired by virtual memory technology, vLLM [17] proposes paged attention to alleviate the memory waste and inefficiency problems of LLM at runtime. Still, it is also aimed at batch-based online service scenarios. In contrast, FlexInfer can be applied to any edge devices and optimized for on-device local inference.

5.2 Offloading for On-device Inference

For inference on edge devices, several offloading-based methods that do not rely on GPUs have been proposed [2, 8, 35, 47]. Llama.cpp [8] is an LLM inference engine implemented in C/C++. It supports running in multiple environments and uses mmap to achieve offloading on SSD. However, its performance is unsatisfactory under memory constraints and lacks scalability. Sparsity-based selective model loadings such as LLMFlash [2] and PowerInfer [35, 47] significantly improve inference performance by reducing the IO size at the algorithm level. However, their performance improvement depends on sparsity and may impact model capabilities. In contrast, FlexInfer can be directly applied to any transform-based model while maintaining full capabilities. Other inference optimization methods, such as model compression [19, 46, 50], model quantization [2, 23, 35, 43, 47], speculative decoding [18, 45], etc., are orthogonal to the method proposed in this paper.

6 Conclusion

Large Language Models (LLMs) pose significant challenges for local inference in resource-constrained environments. This paper addresses these challenges by introducing FlexInfer, a novel framework that combines asynchronous prefetching, balanced memory locking, and flexible tensor preservation to optimize offloading-based LLM inference. FlexInfer minimizes I/O overhead and maximizes memory efficiency under diverse user-specified resource budgets. Extensive experiments demonstrate its superiority over existing methods, making it a practical solution for deploying LLMs locally. This work paves the way for more flexible, efficient, and accessible LLM applications in privacy-sensitive and local scenarios.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. arXiv:2305.13245 [cs.CL] https://arxiv.org/abs/2305.13245

[2] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2024. LLM in a flash: Efficient Large Language Model Inference with Limited Memory. arXiv:2312.11514 [cs.CL] https://arxiv.org/abs/2312.11514

[3] Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. 2022. DeepSpeed- Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15. https://doi.org/10.1109/SC41404.2022.00051

[4] Lihu Chen and Gaël Varoquaux. 2024. What is the Role of Small Models in the LLM Era: A Survey. arXiv:2409.06857 [cs.CL] https://arxiv.org/abs/2409.06857

[5] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 16344–16359. https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf

[6] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024). https://doi.org/10.48550/ARXIV.2407.21783 arXiv:2407.21783

[8] Georgi Gerganov. 2024. ggerganov/llama.cpp: Port of Facebook's LLaMA model in C/C++. https://github.com/ggerganov/llama.cpp.

[9] Junhui He, Shangyu Wu, Weidong Wen, Chun Jason Xue, and Qingan Li. 2024. CHESS: Optimizing LLM Inference via Channel-Wise Thresholding and Selective Sparsification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 18658–18668. https://doi.org/10.18653/v1/2024.emnlp-main.1038

[10] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. arXiv:2305.02301 [cs.CL] https://arxiv.org/abs/2305.02301

[11] Lianming Huang, Shangyu Wu, Yufei Cui, Ying Xiong, Xue Liu, Tei-Wei Kuo, Nan Guan, and Chun Jason Xue. 2024. RAEE: A Training-Free Retrieval-Augmented Early Exiting Framework for Efficient Inference. *CoRR* abs/2405.15198 (2024). https://doi.org/10.48550/ARXIV.2405.15198 arXiv:2405.15198

[12] Wei Huang, Xingyu Zheng, Xudong Ma, Haotong Qin, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. An empirical study of LLaMA3 quantization: from LLMs to MLLMs. *Visual Intelligence* 2, 1 (Dec. 2024). https://doi.org/10.1007/s44267-024-00070-x

[13] HuggingFace. 2022. Hugging face accelerate. https://huggingface.co/docs/accelerate/index.

[14] Paras Jain, Ajay Jain, Aniruddha Nrusimha, Amir Gholami, Pieter Abbeel, Joseph Gonzalez, Kurt Keutzer, and Ion Stoica. 2020. Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 497–511. https://proceedings.mlsys.org/paper_files/paper/2020/file/0b816ae8f06f8dd3543dc3d9ef196cab-Paper.pdf

[15] Cheng Ji, Li-Pin Chang, Liang Shi, Congming Gao, Chao Wu, Yuangang Wang, and Chun Jason Xue. 2017. Lightweight Data Compression for

Mobile Flash Storage. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 183 (Sept. 2017), 18 pages. https://doi.org/10.1145/3126511

[16] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG] https://arxiv.org/abs/2001.08361

[17] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles* (Koblenz, Germany) *(SOSP '23)*. Association for Computing Machinery, New York, NY, USA, 611–626. https://doi.org/10.1145/3600006.3613165

[18] Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning* (Honolulu, Hawaii, USA) *(ICML'23)*. JMLR.org, Article 795, 13 pages.

[19] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: Activation-aware Weight Quantization for On-Device LLM Compression and Acceleration. In *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. De Sa (Eds.), Vol. 6. 87–100. https://proceedings.mlsys.org/paper_files/paper/2024/file/42a452cbafa9dd64e9ba4aa95cc1ef21-Paper-Conference.pdf

[20] Linux. 2024. cgroups - Linux control groups. https://man7.org/linux/man-pages/man7/cgroups.7.html.

[21] Linux. 2024. mmap, munmap - map or unmap files or devices into memory. https://man7.org/linux/man-pages/man2/mmap.2.html.

[22] Linux. 2024. taskset - set or retrieve a process's CPU affinity. https://man7.org/linux/man-pages/man1/taskset.1.html.

[23] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 22137–22176. https://proceedings.mlr.press/v202/liu23am.html

[24] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, and Beidi Chen. 2023. Deja Vu: Contextual Sparsity for Efficient LLMs at Inference Time. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 22137–22176. https://proceedings.mlr.press/v202/liu23am.html

[25] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Christopher Leung, Jiajie Tang, and Jiebo Luo. 2024. LLM-Rec: Personalized Recommendation via Prompting Large Language Models. arXiv:2307.15780 [cs.CL] https://arxiv.org/abs/2307.15780

[26] Yu Mao, Weilan Wang, Hongchao Du, Nan Guan, and Chun Jason Xue. 2024. On the Compressibility of Quantized Large Language Models. arXiv:2403.01384 [cs.LG] https://arxiv.org/abs/2403.01384

[27] NVIDIA. 2023. FasterTransformer. https://developer.nvidia.com/nvidia-triton-inference-server.

[28] OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. 2024. OpenAI o1 System Card. arXiv:2412.16720 [cs.AI] https://arxiv.org/abs/2412.16720

[29] Shishir G. Patil, Paras Jain, Prabal Dutta, Ion Stoica, and Joseph Gonzalez. 2022. POET: Training Neural Networks on Tiny Devices with Integrated Rematerialization and Paging. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 17573–17583. https://proceedings.mlr.press/v162/patil22b.html

[30] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently Scaling Transformer Inference. arXiv:2211.05102 [cs.LG] https://arxiv.org/abs/2211.05102

[31] PrivateGPT 2023. PrivateGPT. https://github.com/zylon-ai/private-gpt.

[32] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 551–564. https://www.usenix.org/conference/atc21/presentation/ren-jie

[33] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 31094–31116. https://proceedings.mlr.press/v202/sheng23a.html

[34] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 31094–31116. https://proceedings.mlr.press/v202/sheng23a.html

[35] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2024. PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU. In *Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles* (Austin, TX, USA) *(SOSP '24)*. Association for Computing Machinery, New York, NY, USA, 590–606. https://doi.org/10.1145/3694715.3695964

[36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *CoRR* abs/2302.13971 (2023). https://doi.org/10.48550/ARXIV.2302.13971 arXiv:2302.13971

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/ARXIV.2307.09288 arXiv:2307.09288

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[39] Linnan Wang, Jinmian Ye, Yiyang Zhao, Wei Wu, Ang Li, Shuaiwen Leon Song, Zenglin Xu, and Tim Kraska. 2018. Superneurons: dynamic GPU memory management for training deep neural networks. *SIGPLAN Not.* 53, 1 (Feb. 2018), 41–53. https://doi.org/10.1145/3200691.3178491

[40] Weilan Wang, Yu Mao, Tang Dongdong, Du Hongchao, Nan Guan, and Chun Jason Xue. 2024. When Compression Meets Model Compression: Memory-Efficient Double Compression for Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 16973–16983. https://doi.org/10.18653/v1/2024.findings-emnlp.988

[41] Xiaohui Wang, Ying Xiong, Yang Wei, Mingxuan Wang, and Lei Li. 2021. LightSeq: A High Performance Inference Library for Transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, Young-bum Kim, Yunyao Li, and Owen Rambow (Eds.). Association for Computational Linguistics, Online, 113–120. https://doi.org/10.18653/v1/2021.naacl-industry.15

[42] Shangyu Wu, Hongchao Du, Ying Xiong, Shuai Chen, Tei wei Kuo, Nan Guan, and Chun Jason Xue. 2025. EvoP: Robust LLM Inference via Evolutionary Pruning. arXiv:2502.14910 [cs.CL] https://arxiv.org/abs/2502.14910

[43] Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. 2023. Flash-LLM: Enabling Cost-Effective and Highly-Efficient Large Generative Model Inference with Unstructured Sparsity. arXiv:2309.10285 [cs.DC] https://arxiv.org/abs/2309.10285

[44] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and Scalable On-device Large Language Model Inference. arXiv:2309.04255 [cs.NI] https://arxiv.org/abs/2309.04255

[45] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and Scalable On-device Large Language Model Inference. arXiv:2309.04255 [cs.NI] https://arxiv.org/abs/2309.04255

[46] Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. OneBit: Towards Extremely Low-bit Large Language Models. arXiv:2402.11295 [cs.CL] https://arxiv.org/abs/2402.11295

[47] Zhenliang Xue, Yixin Song, Zeyu Mi, Xinrui Zheng, Yubin Xia, and Haibo Chen. 2024. PowerInfer-2: Fast Large Language Model Inference on a Smartphone. arXiv:2406.06282 [cs.LG] https://arxiv.org/abs/2406.06282

[48] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*. USENIX Association, Carlsbad, CA, 521–538. https://www.usenix.org/conference/osdi22/presentation/yu

[49] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An Open-Source Small Language Model. arXiv:2401.02385 [cs.CL] https://arxiv.org/abs/2401.02385

[50] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. 2024. Atom: Low-Bit Quantization for Efficient and Accurate LLM Serving. In *Proceedings of Machine Learning and Systems*, P. Gibbons, G. Pekhimenko, and C. De Sa (Eds.), Vol. 6. 196–209. https://proceedings.mlsys.org/paper_files/paper/2024/file/5edb57c05c81d04beb716ef1d542fe9e-Paper-Conference.pdf

[51] Zhe Zhou, Xuechao Wei, Jiejing Zhang, and Guangyu Sun. 2022. PetS: A Unified Framework for Parameter-Efficient Transformers Serving. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 489–504. https://www.usenix.org/

conference/atc22/presentation/zhou-zhe