Extended abstracts: Resource-efficient LLM Inference Serving with Heterogeneous GPUs

Anonymous ECCV 2024 Submission

Paper ID #20

Abstract. In this poster, we present Hels, a heterogeneity-aware LLM inference serving system designed to effectively utilize both computation and memory resources of heterogeneous GPUs. The basic idea of Hels is to leverage the heterogeneity of diverse GPUs by dynamically adjusting the number of attention heads assigned to each GPU, and the proportion of KV caches stored in the memory of each GPU. We have implemented a prototype of Hels on top of the state-of-the-art LLM inference system vLLM. Experimental results with various GPU types demonstrate that Hels can reduce the TPOT (time per output token) by 1.33x and improve overall throughput by up to 1.7x compared to state-of-the-art baselines.

1 Introduction

In recent years, generative large language models (LLMs) have been successfully applied across various domains, including intelligent agents, chatbots, and code generation. As this trend continues, the demand for computing resources to serve these large models has skyrocketed. Service providers, such as OpenAI, typically deploy tens of thousands of high-end GPUs (e.g., Nvidia A100) for LLM inference serving. However, due to the highly dynamic nature of inference requests, over-provisioning too many high-end GPUs will result in low resource utilization, leading to significant resource waste and carbon emissions.

In this poster, we propose to improve the resource and cost efficiency of LLM Inference Serving by exploiting heterogeneous GPUs. Specially, with the rapid upgrade and iteration of GPUs, existing data centers typically house different GPUs, such as Nvidia's A100, V100, 4090, and 3090 [4]. Different GPUs have varying power efficiencies and capabilities, allowing for strategic allocation of tasks to the most suitable hardware. This approach ensures that each GPU is used to its full potential without being overburdened or underutilized. Additionally, heterogeneous systems facilitate effective load balancing and scalability, allowing for dynamic resource allocation that prevents energy and cost waste [5].

To maximize the efficiency of using heterogeneous GPUs for LLM inference serving, we present Hels, a heterogeneity-aware LLM inference serving system designed to effectively utilize both computation and memory resources of diverse GPUs. The basic idea of Hels is to leverage the heterogeneity of the GPUs by dynamically adjusting the number of attention heads assigned to each GPU, and the proportion of KV caches stored in the memory of each GPU. We have implemented a prototype of Hels on top of the state-of-the-art LLM inference system vLLM [3]. Experimental results with various GPU types demonstrate that Hels can reduce the TPOT (time per output token) by 1.33x and improve overall throughput by up to 1.7x compared to state-of-the-art baselines.

2 Motivation

When deploying LLM inference serving on clusters with heterogeneous GPUs, existing approaches fall into two categories: heterogeneity-ignorant and heterogeneity-aware. In heterogeneity-ignorant approaches, such as vLLM [3], all GPUs are treated as if they have identical computational and memory capabilities, and model parameters are divided equally among them. In contrast, heterogeneity-aware approaches, like HexGen [2], partition LLM models into multiple shards with varying numbers of heads (tensor parallelism) or layers (pipeline parallelism) and assign these shards to GPUs based on their memory or computational capacities.

However, both approaches have significant limitations. Heterogeneity-ignorant approaches face challenges when GPUs with limited memory cannot store as many key-value (KV) caches as those with more memory, leading to reduced batch sizes and limited throughput because the memory of all GPUs isn't fully utilized. For example, comparing an RTX 3090 GPU and a Tesla A100 GPU, which have 24GB and 80GB of HBM respectively, when the 3090's memory is fully utilized, the A100's utilization can drop to below 30%.

Table 1: The memory capacity and inference time across different GPUs

Device	Memory	Time (Prefill)	Time (Decode)
A100	80 GB	0.06s	0.0097s
3090	24 GB	0.147s	0.0143s
P100	12 GB	1.47s	0.077s

On the other hand, heterogeneity-aware approaches can result in a mismatch between computation and memory capacities, leading to underutilization of either computation or memory resources. A simple experiment comparing an RTX 3090, a Tesla P100, and a Tesla A100 GPU showed that while the A100 offers a 24.5× acceleration in the prefill phase and a 7.9× acceleration in the decode phase compared to the P100, its memory capacity isn't 7.9× larger than the P100's. In contrast, the 3090 increases decode latency by less than 1.5× while using less than 40% of the A100's memory footprint. Additionally, varying communication overheads across GPUs lead to imbalanced request processing latencies.

A practical approach to harmonize the computational and memory capacities of heterogeneous GPUs involves balancing memory usage with computational resources. This is achieved by storing portions of the key-value (KV) caches in storage and dynamically recomputing the remaining segments as needed. While this strategy increases inference times slightly, it optimizes resource utilization without significantly compromising performance. GPUs with higher computational capabilities but lower memory capacity can load more model parameter shards and reduce memory requirements by recomputing, thus enhancing resource efficiency and overall throughput.

3 Dynamical Head Assignment and KV Cache Allocation

To leverage the heterogeneity of the GPUs, we dynamically adjust the number of attention heads assigned to each GPU, and the proportion of KV caches stored in the memory of each GPU. The objective of dynamical head assignment and KV cache allocation is to achieve a balanced latency among heterogeneous GPUs to minimize the overall latency. Specially, given a numbers of R user requests and a set of $\mathcal{N} = \{1, 2, \cdots, N\}$ GPUs, the problem of joint head assignment and KV cache allocation can be formulated as follows:

$$\min \max_{j \in \mathcal{N}} \ f_j(R, P_j, H_j), \ \text{s.t.} \ 0 \le P_j \le 1, \sum_{j=1}^{N} P_j = 1, \sum_{j=1}^{N} H_j = TH, \ H_j \in \mathbb{N}. \quad (1)$$

Here H_j denotes the number of attention heads assigned to GPU j and TH denotes the total number of heads in the current model. P_j denotes the proportion of KV caches stored the memory of GPU j. Function f_j is the inference latency and communication latency of GPU j. According to our empirical measurements, f_j can be approximated via a linear regression model.

To reduce the search space and quickly decide a near-optimal solution for the above problem, we take an iterative step-wise attention heads assignment scheme. Specifically, in each iteration, we first calculate the KV cache allocation to each GPU according to their memory capability C_j. Then we obtain the latency based on the latency model f_j, if the overall latency no longer decreases, we obtain an near optimal head assignment KV cache allocation. Otherwise, we allocate more heads to the GPUs with lower latency, and reclaims heads from the ones with higher latency. According to the experiments, this algorithm has linear time complexity and can be terminated in milliseconds.

4 Preliminary Evaluation

4.1 Evaluation Setup

We have implemented Hels on the top of the widely-adopted LLM serving systems vLLM [3] with 2KLOC of Python in approximately. We leverage a local heterogeneous cluster consisting of the following hosts: two hosts with four A100 GPUs, two hosts with two NVIDIA 3090 GPUs each, and a host with two P100 GPUs. To emulate real-world LLM serving, we generated inference request workloads by leveraging characteristics from the OpenChat datasets [1], where the prompt length of requests exhibits substantial diversity and within a specific distribution. We also evaluate Hels across a variety of LLM models under different hardware configurations, as listed in Table 2. All LLM serving models were

Table 2: Models used in experiments and corresponding cluster configurations

Model	GPU Configuration in experiments
Llama-2-13b	A100*2+P100*2
OPT-30b	A100*2+3090*2

Table 3: TPOT (seconds) of Llama-2-13b

LLM system	bs=32	bs=64	bs=128	bs=256
Hels	0.0350	0.038	0.048	0.079
vLLM	0.0568	0.0738	OOM	OOM
HexGen	0.0352	0.039	0.053	0.085

executed using PyTorch 2.1.2, CUDA 12.4, and NCCL 2.18.1. We compare Hels with two SOTA baselines, vLLM [3] and HexGen [2].

Table 4: TPOT (seconds) of OPT-30b (bs=batch size)

LLM system	bs=32	bs=64	bs=128	bs=256
Hels	0.0444	0.0525	0.0715	0.118
vLLM	0.0508	0.0738	OOM	OOM
HexGen	0.0482	0.0643	0.0934	OOM

4.2 Evaluation Results

Time Per Output Token(TPOT): Table 3 shows that Hels consistently outperforms the baselines under various configurations, achieving a reduction on TPOT by up to 1.32x compared with HexGen and a remarkable 1.94x acceleration compared with vLLM. This is because that Hels leverages an advanced dynamic KV caches storage strategy to fully utilize the memory and computation capability of all servers, which not only reduces the TPOT in comparison to the baselines, but also increases maximum number of requests can be served in each iteration.

Table 5: Normalized throughput of OPT-30b

LLM system	Median Throughput	P95 Throughput
Hels	1	1
vLLM	0.54	0.41
HexGen	0.72	0.58

Throughput: We assess the throughput of Hels and baselines using a long-term trace with time-varying arrival patterns, containing 2000 requests. As shown in Table 5, Hels achieves an improvement by up to 2.43x and 1.72x in overall throughput compared with vLLM and HexGen. This performance gain mainly attributes to the comprehensive utilization of memory capability across the clusters, enabling Hels to host more requests to saturate the computation resource.

References

1. Cohan, A., Dernoncourt, F., Kim, D.S., Bui, T., Kim, S., Chang, W., Goharian, N.: A discourse-aware attention model for abstractive summarization of long documents. arXiv preprint arXiv:1804.05685 (2018) 3
2. Jiang, Y., Yan, R., Yao, X., Zhou, Y., Chen, B., Yuan, B.: Hexgen: Generative inference of large language model over heterogeneous environment. In: Forty-first International Conference on Machine Learning 2, 4
3. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., Gonzalez, J., Zhang, H., Stoica, I.: Efficient memory management for large language model serving with pagedattention. In: Proceedings of the 29th Symposium on Operating Systems Principles. pp. 611–626 (2023) 2, 3, 4
4. Weng, Q., Xiao, W., Yu, Y., Wang, W., Wang, C., He, J., Li, Y., Zhang, L., Lin, W., Ding, Y.: Mlaas in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In: 19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022 1
5. Yang, Z., Wu, Z., Luo, M., Chiang, W., Bhardwaj, R., Kwon, W., Zhuang, S., Luan, F.S., Mittal, G., Shenker, S., Stoica, I.: Skypilot: An intercloud broker for sky computing. In: 20th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2023, Boston, MA, April 17-19, 2023 1