ROBIN: A Frugal Mixture of Experts based LLM Inference System

Rajveer Bachkaniwala
rr@gatech.edu

Ziyuan Cao
caoziyuan@gatech.edu
Georgia Institute of Technology

Aniruddha Mysore
animysore@gatech.edu

1 Motivation

Recently, systems for LLM inference serving at a low cost by leveraging spot instances in the cloud have gained attention as well as adoption. But, we found that previous systems lack support for LLMs based on Mixture of Experts (MoE) as the model architecture design choice. MoE have proved to provide competitive performance on inference tasks with low latency and high throughput while consuming less compute and energy in comparison to GPT models. We propose, **ROBIN**, an MoE based LLM inference system serving at a low cost by leveraging spot instances and accelerator heterogeneity in cloud. Our system trades model accuracy for a low cost inference system and an uptime of at least 99% (SLO).

2 Related Work

To our knowledge, no prior work/system supports serving **MoEs based LLM inference** with all of the following – **(a) spot instances**, **(b) heterogeneous accelerators**, and **(c) 99% SLO**. Here, we differentiate **ROBIN** by providing details of systems focusing on subsets of the problem:

- **LLM Inference for pre-emptible compute:** Recent work on fault-tolerant inference like [2] would support pre-emptible cloud VMs as well, however they do not aim to provide any SLO guarantees on latency. SpotServe [8] is a system specialized for LLM Inference on spot instances. However it does not currently support hard SLO guarantees, which our system aims to do. SkyServe [12] is an LLM-inference system that allows the developer to make trade-offs between latency and cost, and achieves cost-savings by using spot instances when specified.

- **General LLM Inference systems:** State-of-the-art LLM inference systems like FasterTransformer [9] and Orca [14] require a fixed compute cluster and have not been evaluated on Spot VMs.

- **MoE Inference Systems** MoE-based LLMs have been growing in popularity since they offer high-throughput in token generation. However the large parameter size when considering the system as a whole, has made inference expensive since multiple GPUs are needed just to load the model. Many systems have proposed solutions regarding this problem namely – quantization [3], offloading experts to CPU memory/SSD [3], tracing expert activation [13], expert caching and prefetching [13], and inferring experts only on CPU due to low cost of transferring activations [6]. But, these systems have not focused on multi-GPU setting solutions due to cost of acquiring more GPUs.

- **LLM training on pre-emptible compute:** State-of-the-art training solutions over SpotVMs exist, and while we can draw ideas from them, training and inference have fundamentally different problem characteristics. Varuna [1] allows training LLMs over cost-effective low-priority VMs that are preemptible and 5x cheaper. Their contributions include a new micro-batch scheduling algorithm, an algorithm to dynamically change pipeline depth and data-parallelism dimensions, and an automatic model partitioning technique. Bamboo [11] studies resilient training of LLMs over preemptible instances by adding redundant computations in the pipeline parallel schedule. However, latency is not the focus of those systems.

- **Spot-based solutions for non-LLM use cases:** Mark [15] explores using on-demand, spot, and burstable serverless compute to serve DNN models cheaply while maintaining latency SLOs. Cocktail [4] serves ensemble models over transient VMs while navigating tradeoffs between accuracy, cost, and latency during model selection. Neither of these SOTA systems serve models larger than 500M in size, and hence are not applicable to our chosen use case of LLMs since LLMs serving often requires model and pipeline parallelism.

3 System challenges

The key feature of MoE is sparse activation which leads to low compute and high throughput. The MoE model in its totality is large in terms of parameter size. Fortunately, only a select few experts are chosen for each token based on the router which means only a few parameters are activated. Unfortunately, the subset of experts chosen for a token are undetermined until after the activations are processed by the router. Since, the original MoE model is large it cannot fit all experts in the GPU memory meaning if the chosen experts are not present in the GPU memory a transfer cost is incurred causing difficulty in scaling. If we consider a setting where we use multiple spot-instances to load all experts, our system will need to handle the case of pre-emption in addition to high infrastructure cost. A naive migration based approach may cause an SLO violation. Our challenge is to design a system such that the experts are available just-in-time for the activations to be processed while also handle pre-emption in a manner that prevents significant disruption of the requests being processed.

4 High-Level Approach

We build upon the open-source system proposed in SpotServe [8], which is based on FasterTransformer [9]. We modify the inference engine to serve a Mixture-of-Expert model and hypothesize that while the modified SpotServe system will offer strong performance when evaluated against the *rerouting* and *reparallelization* baselines [8], it will be unable to meet a 99% uptime target for realistic request arrival patterns. These results will be a good example of why naively using SpotServe is insufficient.

ROBIN trades accuracy to the meet the SLO target in spot instance setting. MoE models can consist of several experts which are selectively used during inference. Even if some experts were not available at a given instance, say, when some spot instances were taken away, we claim that **ROBIN** can still do inference, albeit not as accurately as top-k experts that the MoE model selected. With this relaxed constraint, the problem of maintaining latency becomes tractable, but we believe by no means trivial. There are a few techniques we can leverage to get around this problem which we detail ahead.

4.1 Predicting Expert Selection

MoE Pre-gating [5] is a technique to predict the top-k experts in the previous encoder/decoder block's expert execution, so that if that necessary experts can be prefetched just-in-time from CPU memory/SSD. This technique can be used to hide a situation where spot-instances were taken away, by speculatively swapping-in the now deallocated experts from CPU memory on the remaining instances. The observed downside of Pre-gating is performance degradation when k increases [5].

4.2 Trading Top k Experts Dynamically

We leverage the Pre-gating technique mentioned previously for this approach. For instance, if k is large such that prefetching degrades performance then we do a best effort approach where we prefetch experts $< k$ at the cost of model accuracy. This technique would allow us to meet the SLO requirement.

We would like to briefly mention another approach that we considered. In this approach, assume for a give k, only part of them are resident on GPU memory in which case if the number of resident experts is greater than half of k then we only use the desired resident ones otherwise we fetch from CPU memory/SSD. We believe this approach to be inferior than the previous approach so we do not pursue it.

5 Evaluation Metric

The goal is to evaluate **ROBIN** under diverse load conditions and pre-emption scenarios with respect to on-demand instance based LLM inference system. A robust metric to check the performance of **ROBIN** with state of the art is E2E latency and throughput. We plan to use vLLM [7] and DeepSpeed-MoE [10] as the SOTA systems for comparison. To study the robustness of **ROBIN** using spot instances, we also want to evaluate on more general traces that also represents locality information. This is non-trivial - we need to find or collect data about VM availability, model increased latency in the case of far-off instances and simulate it in the system during trace replayment.

References

[1] Athlur, S., Saran, N., Sivathanu, M., Ramjee, R., and Kwatra, N. Varuna: scalable, low-cost training of massive deep learning models. In *Proceedings of the Seventeenth European Conference on Computer Systems* (New York, NY, USA, 2022), EuroSys '22, Association for Computing Machinery, p. 472–487.

[2] Borzunov, A., Ryabinin, M., Chumachenko, A., Baranchuk, D., Dettmers, T., Belkada, Y., Samygin, P., and Raffel, C. Distributed inference and fine-tuning of large language models over the internet. In *Thirty-seventh Conference on Neural Information Processing Systems* (2023).

[3] Eliseev, A., and Mazur, D. Fast inference of mixture-of-experts language models with offloading, 2023.

[4] Gunasekaran, J. R., Mishra, C. S., Thinakaran, P., Sharma, B., Kandemir, M. T., and Das, C. R. Cocktail: A multidimensional optimization for model serving in cloud. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)* (Renton, WA, Apr. 2022), USENIX Association, pp. 1041–1057.

[5] Hwang, R., Wei, J., Cao, S., Hwang, C., Tang, X., Cao, T., and Yang, M. Pre-gated moe: An algorithm-system co-design for fast and scalable mixture-of-expert inference, 2023.

[6] Kamahori, K., Gu, Y., Zhu, K., and Kasikci, B. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models, 2024.

[7] Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles* (2023).

[8] Miao, X., Shi, C., Duan, J., Xi, X., Lin, D., Cui, B., and Jia, Z. Spotserve: Serving generative large language models on preemptible instances, 2023.

[9] NVIDIA. Fastertransformer: High-performance inference library for transformers, January 2023. GitHub repository.

[10] Rajbhandari, S., Li, C., Yao, Z., Zhang, M., Aminabadi, R. Y., Awan, A. A., Rasley, J., and He, Y. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning* (17–23 Jul 2022), K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162 of *Proceedings of Machine Learning Research*, PMLR, pp. 18332–18346.

[11] Thorpe, J., Zhao, P., Eyolfson, J., Qiao, Y., Jia, Z., Zhang, M., Netravali, R., and Xu, G. H. Bamboo: Making preemptible instances resilient for affordable training of large dnns. *CoRR abs/2204.12013* (2022).

[12] Xia, T., Wu, Z., Mao, Z., and Yang, Z. Introducing skyserve: 50 Accessed: 2024-02-23.

[13] Xue, L., Fu, Y., Lu, Z., Mai, L., and Marina, M. Moe-infinity: Activation-aware expert offloading for efficient moe serving, 2024.

[14] Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)* (Carlsbad, CA, July 2022), USENIX Association, pp. 521–538.

[15] Zhang, C., Yu, M., Wang, W., and Yan, F. MArk: Exploiting cloud services for Cost-Effective, SLO-Aware machine learning inference serving. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)* (Renton, WA, July 2019), USENIX Association, pp. 1049–1062.