Qeios

Research Article

Towards More Economical Context-Augmented LLM Generation by Reusing Stored KV Cache

Hanchen Li[1], Yuhan Liu[1], Yihua Cheng[1], Kuntai Du[1], Junchen Jiang[1]

1. University of Chicago, United States

Across large language model (LLM) applications, we observe an emerging trend for reusing KV caches to save the prefill delays of processing repeated input texts in different LLM inputs. This has led to a broad design space, including colocating stored KV caches with (or close to) GPUs to various KV cache compression. However, a key question remains unanswered: can these delay reductions also be economically favorable? Specifically, we ask whether a developer can use public cloud services to store precomputed KV caches and reuse them to save delay without incurring more costs in terms of compute, storage, and network. To answer this question, we propose an validated analytical model for the cloud cost (in compute, storage, and network) of storing and reusing KV caches based on various workload parameters, such as reuse frequency, generated text lengths, model sizes, etc. Preliminary results show that KV cache reusing is able to save both delay and cloud cost across a range of workloads with long context. And we call more efforts on building more economical context augmented LLM by KV cache reusing.

Hanchen Li and Yuhan Liu equally contributed to this work.

1. Introduction

Nowadays, a wide range of applications (e.g., document-based question answering) often use large language models (LLMs) after prepending the LLM inputs with *long contexts* (e.g., related documents) that may contain thousands of tokens or more. In this paper, we refer such long prefixes as contexts.

Prior work found that different user requests may need to use the same context[1]. Therefore, the computation results of the context (in the format of KV caches), once computed and cached in the GPU,

can be reused across those jobs that use the same context[2][3]. For example, two requests, "What was the highest source of revenue of Apple in the last year?" and "Which supplier for Apple offers the lowest average price?", may need to use the same annual fiscal report as the context.

However, the design challenge of such KV cache reusing scheme, is *where to store the pre-computed KV cache of the context?* One potential design is to store the KV cache in the GPU[2][3], which is feasible when there are only limited number of contexts that the user will access (e.g., a dedicated server that answers the question about a specific software), but is less favorable when a wide range of context may be accessed, as the GPU will not have sufficient GPU memory to store the KV cache of all the contexts.

An alternative solution is to store the KV cache of the context in a dedicated storage server and transfer the KV cache into GPU memory when new jobs requesting contexts arrive. Acknowledging that such an approach incurs extra transfer delay and storage cost, we argue that such an approach is favorable compared with text recomputation in terms of both end-to-end delay and economic cost. By building an analytical model for developers to compare service costs given their workload pattern and cloud service pricing policy, we show that reusing KV cache from storage is more economical than text recomputation. Moreover, we validate this result by simulation under various workloads. This proves that the end-to-end delay and economic cost of serving LLM inference jobs with KV cache are consistently lower than without storing KV cache. We hope this poster sparks more discussions toward realizing the full potential of such systems.
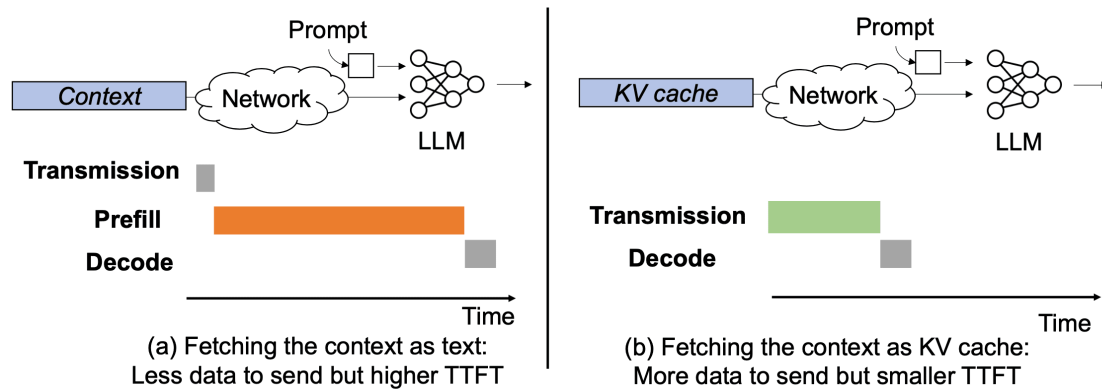

Figure 1. Illustration of text recomputation and KV cache reusing.

2. Cost Modeling

Analytical Model: In order to compare the operation costs of text recomputation and KV cache reusing, we built an analytical model to measure their costs. For the first text recomputation pipeline, the inference serving cost (*i.e.*, GPU compute) for one request could be estimated by the GPU use time per request times the number of requests using the context within a time period. The breakdown cost can be expressed by the following equation:

$$C_{text} = C_{GPU} \times N\left[T_{\text{prefill}}(L_{\text{context}} + L_{\text{prompt}}) + T_{\text{decode}}(L_{\text{output}})\right],$$

where C_{GPU} is the cost of the GPU instance per hour, N is number of requests querying the same context, $L_{context}$ is the number of context tokens, L_{prompt} is the token length of the actual prompt, and L_{output} is the number of generated tokens. $T_{prefill}$ and T_{decode} are time (hour) spent on the computation job given specific token numbers.

For the context reusing pipeline, we separate the cost into three parts: inference, context storage, and transmission cost:

$$C_{KV} = \overbrace{C_{GPU} \times \{N\left[T_{\text{decode}}(L_{\text{output}}) + T_{\text{prefill}}(L_{\text{prompt}})\right] + T_{\text{prefill}}\}}^{\text{GPU inference cost}}$$

$$+ \overbrace{C_{\text{storage}} \times S_{storage}(L_{\text{context}}) \times T}^{\text{Storage cost}}$$

$$+ \overbrace{C_{\text{transmission}}(S_{storage}(L_{\text{context}}), SLO)}^{\text{Transmission cost}},$$

where C_{storage} is the storage cost per hour$\times$GB, T is the period of serving, S_{storage} is the size of the KV cache, SLO is the service level objective for time to first token, and C_{storage} is the cost per hour to maintain the transmission link that satisfies SLO for loading the KV cache.

Insights: By analyzing the model with real pricing data from AWS, we find that Reusing KV cache could be more economical as long as the context is reused more than once per hour. And that the storage cost is usually only a minimal portion of the total cost. Take Llama-7B model as an example. A context of length 10K-token requires a storage space of 5.2 GB. With Amazon Elastic Block Storage(EBS) that could be directly mounted onto instance, the highest tier storage with 4GB/S throughput, called io2, costs 0.125 per GB $\times$ month [4] Since our IO operations are not frequent, we do not need to pay much extra provisioned IOPS fee. Within the period of one hour LLM serving. The total storage and transmission cost spent this hour for this 10K-token context is $ 8.8e^{-4\,(1)}$. On the other hand, the GPU inference cost for prefilling is

around $6e^{-3}$ for a single request[^2]. This is already more than 7 times larger than the fixed cost of storage during this hour. And if more requests reuse this piece of context in this hour, this fixed cost in reusing KV cache will be negligible in total cost. We can simplify the ratio between C_{text} and C_{KV}:

$\frac{C_{\text{text}}}{C_{\text{KV}}} \approx 1 + \frac{N-1}{N} \frac{T_{\text{prefill}}(L_{\text{context}})}{T_{\text{decode}}(L_{\text{output}}) + T_{\text{prefill}}(L_{\text{prompt}})}$. This demonstrates that when context is long while the prompt and output are short, such as in many article Q&A tasks[1], the savings of loading KV cache from storage could be significant.

3. Preliminary Results

Evaluation setup: In the following experiment, we run Llama-7B model on an Amazon's EC2 p3.8xlarge instance (4 V100 GPUs). We apply HuggingFace's model parallelism to run inference on four GPUs. The dataset is TriviaQA[1], which contains 200 pieces of different contexts. We assume each context is used five times and vary two workload parameters, the input lengths and output lengths.
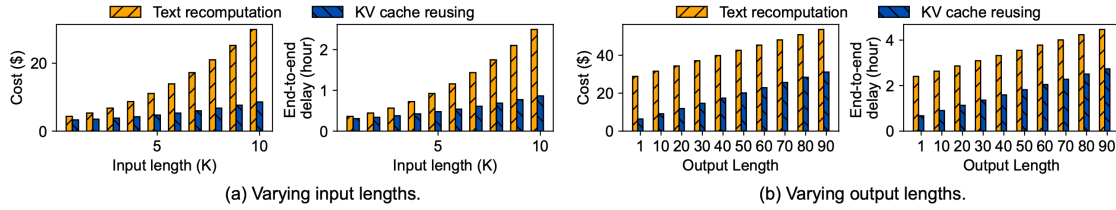

(a) Varying input lengths.   (b) Varying output lengths.

Figure 2. The cost and end-to-end delay for Llama-7B by varying input lengths.

Takeaways:

- When varying the input length from 1–10K, as shown in Figure 2(a), using KV cache reuse can significantly lower both the end-to-end delay (by 1.1 to 2.9 $\times$) and the cost of using cloud services (by about 1.3 to 3.6 $\times$). The benefits of KV cache reuse become more significant as the length of the input increases. For shorter inputs (like 1,000 units), the time saved by skipping the prefilling delay does not compromise the delay it takes to load the KV cache.

- In Figure 2(b) where the output lengths are varied from 1 to 100 tokens, the delay saving ranges from 1.6–3.5$\times$ and the cost saving ranges from 1.7–4.5$\times$. We observe that the longer the output is, the less saving KV cache reusing can bring. The insight is that KV cache reusing is only able to save delay for

the first token's generation, thus, the longer the output is, the more likely the saving will be amortized.

Footnotes

[1] io2 storage cost per hour $\times size = 0.00017 * 5.2 = 0.00088$

[2] Single V100 GPU cost per hour $\times \frac{1h}{3600s} \times T_{prefill}(10K) = \$3 \times \frac{1}{3600} \times 7 = 0.0058$.

References

1. [a, b, c]Joshi M, Choi E, Weld DS, Zettlemoyer L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension [Preprint]. arXiv. 2017. Available from: https://arxiv.org/abs/1705.03551

2. [a, b]Kwon W, Li Z, Zhuang S, Sheng Y, Zheng L, Yu CH, et al. Efficient memory management for large language model serving with PagedAttention. In: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles. 2023.

3. [a, b]Anonymous. ChunkAttention: Efficient attention on KV cache with chunking sharing and batching [Internet]. 2024. Available from: https://openreview.net/forum?id=9k27IITeAZ

4. [^]Amazon Web Services. Amazon EBS pricing [Internet]. Available from: https://aws.amazon.com/ebs/pricing/

Declarations