Optimizing Cache Memory Usage Methods for Chat LLM-models in PaaS Installations

Mikhail. M. Rovnyagin
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russia
m.rovnyagin.2015@ieee.org

Dmitry M. Sinelnikov
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russia
dsinelnikov96@gmail.com

Artem A. Eroshev
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russia
aeroshev@mail.ru

Tatyana A. Rovnyagina
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russia
tanya.plys@bk.ru

Alexander V. Tikhomirov
National Research Nuclear University MEPhI
(Moscow Engineering Physics Institute)
Moscow, Russia
avtikhomirov@mephi.ru

***Abstract* — Recently, LLM models have become widespread in industry. They are used as the basis for voice assistants, troubleshooting systems, chatbots and much more. The work of LLM is based on the architecture of transformer-type neural networks, where text is supplied as input (for example, text chat), and the model, estimating the character-by-character probability, returns the result also in the form of text. In this case, the model does not retrain. And the context itself is constantly accumulating. This paper proposes two ways to reduce the memory allocated for storing the test chat context. The first method is to periodically launch additional training in order to embed the chat context into the core of the model itself. The article discusses the pros and cons of this approach. The second method is to save in the text chat cache only with those users where this context has already been formed. The article describes the layout for conducting the experiment, provides the results of the experimental study and describes the method for assessing the "maturity" of the chat correspondence context.**

Keywords — ChatGPT, Large Language Model, Transformer-type Neural Network, Context Embedding, Memory Allocation

I. INTRODUCTION

Transformers are a popular machine learning model architecture developed by Google specialists designed for natural language processing. Based on this approach, such well-known large language models as Llama 2 [1], Mistral [2], ChatGPT [3] were developed. The architecture of transformers consists of an encoder block, the so-called encoder, and a decoder block. The input of the encoder block is the source text, which represents text embeddings. The decoder takes as input the result of the encoder's operation and the words that it generated earlier. Another feature that sets this approach apart from other architectures is the use of blocks.

The attention block in the Transformer architecture is a key component that allows the model to focus on different parts of the input data during processing. This is a key feature that differentiates Transformer from previous sequence processing approaches such as recurrent neural networks (RNN) and long short-term memory (LSTM).

Multi-head attention [4] is a mechanism that allows the model to simultaneously pay attention to different subspaces of information in the input data. This is achieved by dividing the attention mechanism into several "heads", each of which can focus on different aspects of information. As a result, the model can better capture the variety of dependencies in the data.

Figure 1 shows a diagram of the transformer architecture from the article [4].

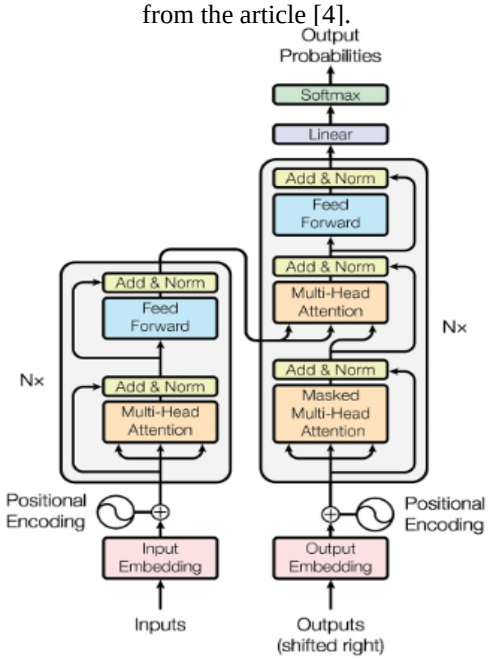

Fig. 1 - Transformers architecture diagram

Analyzing the entire text immediately entails problems in the form of the required memory to store all the words in context. Theoretically, if the technical equipment allowed the use of significant amounts of memory, then another problem would arise related to the operating time on the execution unit.

Due to these architectural features, models built on the basis of Transformers have a limitation in the size of the context [5].

II. RELATED WORKS

Modern SOTA (State-of-the-art) machine learning models consume a fairly significant amount of computing resources, which leads to the study of various optimization methods and load distribution. The paper [6] considers load distribution by slicing a deep network into N:M sparsity.

Each piece is executed on its own device. This approach is interesting because it allows you to run full-fledged models on a pool of small and inexpensive devices, the total cost of which is significantly lower than industrial solutions. However, this solution is not suitable when there are constraints on the availability of a large amount of computing power. In case of competitive access to model resources, it is necessary to store all user correspondence.

In [7], the authors study the problem of training large language models on edge devices and consider several ways to train the model on a distributed system. This study examines the use of DRL (Deep RL) for content caching in the edge nodes of the MEC (Mobile Edge Caching) system. The system has a library of F files of popular content that is requested by all mobile users. The popularity of content is determined based on the likelihood of its requests from all users. It is assumed that content popularity and user preferences remain static over a long period of time.

The authors from another article [8] chose a different approach. Serving high-throughput large language models (LLMs) faces the challenge of managing key-value cache (KV cache) memory, which changes in size quickly and can lead to memory waste. To solve this problem, the PagedAttention algorithm is proposed, based on the principles of virtual memory and pagination. This algorithm is used in a new LLM serving system called vLLM, which can significantly reduce KV cache usage and increase LLM throughput by 2-4 times without increasing latency compared to existing systems. This approach significantly improves the speed of the model's work with data, but does not help reduce memory consumption for numerous queries, which is an important factor when the amount of this resource is limited.

III. Optimization methods

This article focuses on ways to reduce the amount of information stored about user sessions with a correspondence model.

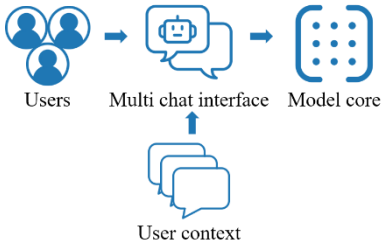

Fig. 2 – Storing user sessions during concurrent access to the model

If the context of communication with the user (Fig. 2) is extremely large, then it is appropriate to apply an approach associated with online learning and adjust the core of the model to the needs of the user. We call this approach "dreams" of the model. The model will no longer store a detailed dialogue with the user, which brings the model's responses to the current state, but once in a certain period of time it will use the dialogue for additional training (fine-tuning).

Also, for correspondence that has passed a certain critical threshold (in terms of its volume), you can ask the model to summarize the correspondence to some volume suitable from the point of view of further storage and thereby reduce the overall costs of storing correspondence contexts.

Finally, as a method for optimizing the memory costs of storing the context of communication with a model, this article considers the method of clearing after the lifetime of the correspondence context (TTL) has expired, taking into account the depth of correspondence between the user and the model.

IV. Context evaluation cache optimization

Since the model spends quite a lot of memory processing the user request, it is necessary to optimize memory use during concurrent access to the model. Thus, we are talking about assessing the usefulness of the context of communication with the model, based on its depth (number of messages) and lifetime.

To study such a scenario for the model, a Telegram bot was developed that transmits user requests to a large language model. The model chosen was Mistral 7B with the Russian-language Saiga dataset as the most modern at the moment. The model was launched on an Nvidia RTX 3050 8 Gb video card with 4 bit quantization [9]. This was necessary because the base model occupies 16 Gb of video memory; using quantization settings, it was possible to reduce consumption to 6 Gb, which leaves 2 Gb for the context of the model. Redis was used to store user correspondence with the model.

V. Experiment

The purpose of this experiment is to study the modes of use of the model by users and find the value of the context depth, which would allow us to determine active and inactive correspondence between users and the model, which would allow us to effectively store hot data in the cache.

To conduct the experiment, two modes of storing the context of correspondence with the user were implemented for the chatbot: a mode of permanent storage of correspondence (inline mode) and a sliding window mode (window mode). The first mode stores all messages received from the user and model; if the memory on the video accelerator runs out, a warning is displayed. The second mode, in case of low memory, throws out the very first message in the window, which allows you to endlessly chat with the model. For the last mode, based on the results of assessing the context depth, the window width will be set.

A group of 7 experts was involved in determining the assessment of context depth. They asked the model questions on various topics. After corresponding with the model, each expert gave his estimate of the number of messages after which the model began to correctly perceive the context of the correspondence with the user.

During the experiment, statistical data on user sessions was collected. Figure 3 shows a graph of the distribution of the number of messages by session.

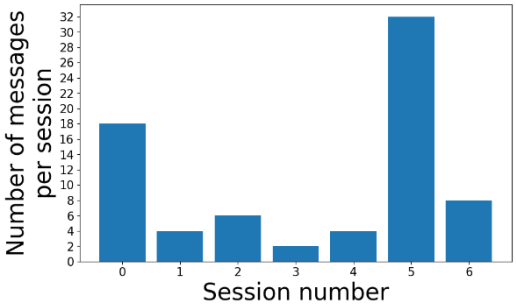

Fig. 3 - Distribution of the number of messages by sessions.

Figure 4 shows the size distribution of all correspondence for each session.

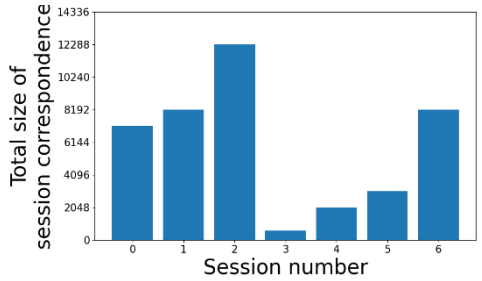

Fig. 4 - Session size distribution in bytes

As you can see, the size of the correspondence does not directly depend on the number of messages. This is due to the fact that the subject matter and wording of requests can vary greatly. The average message size was determined for each session. The median value for all message averages was 512, which we took as the average message size of the model to calculate the efficiency of cache clearing at various thresholds.

To determine an objective assessment of the maximum depth of correspondence in the cache, the method of pairwise comparisons of Thomas Saaty's criteria was used [10]. Its essence is for the expert to give his rating in the range from 1/9 (much worse) to 9 (much superior) for each pair of characteristics. In this way, a wide range of criteria can be assessed objectively. The criteria used were the number of messages in the correspondence at which the expert stated that the model began to understand the context of the correspondence in full. As a result, a table was compiled filled with averaged expert assessments.

TABLE I. PAIRWISE COMPARISONS

Criteria	Msg count = 4	Msg count = 5	Msg count = 7	Msg count = 8	Msg count = 10	Msg count = 12	Msg count = 15	Product	Totals	Normalize totals
Msg count = 4	1	1/3	1/5	1/9	1/7	1/5	1/3	0,0000705	0,255226	0,026506674
Msg count = 5	3	1	1/3	1/7	1/5	1/5	1/3	0,001905	0,408701	0,042445902
Msg count = 7	5	3	1	1/5	1/4	1/4	1/3	0,0625	0,67295	0,069889642
Msg count = 8	9	7	5	1	1/3	1/3	1/3	11,67	1,420434	0,147520008
Msg count = 10	7	5	4	3	1	3	3	3780,00	3,243921	0,336899383
Msg count = 12	5	5	4	3	1/3	1	3	300,00	2,258783	0,23458726
Msg count = 15	3	3	3	3	1/3	1/3	1	9,00	1,368738	0,14215113

The largest of the normalized vectors was the vector for the sign of the number of messages 10. This parameter was adopted by setting the size of the sliding window.

Figure 5 shows the dependence of memory consumption before and after applying a limit on the maximum depth of correspondence, where the red graph is the original data, and the green one is after applying the method of optimizing the storage of correspondence data.

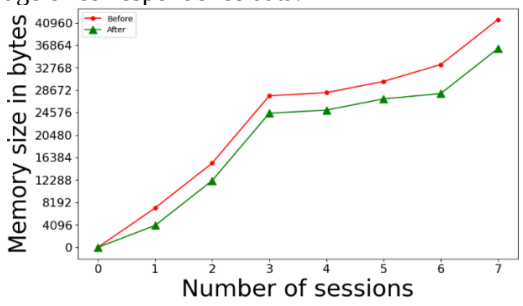

Fig. 5 - Memory efficiency at different thresholds.

VI. CONCLUSION

Thus, this article examined the Transformers model as an object of competitive multi-user access. The problem of complexity of storing the context of correspondence with users is identified and some ways to solve it are given: "dreams" (adjusting the core of the model for specific users), summarizing the context, deleting the context based on its usefulness. The last method was tested experimentally. Based on expert assessments for a specific instance of the LLM model, the context depth was calculated, starting from which the model became quite "specialized." A layout was developed that, after some time, deleted all correspondence except specialized ones (thus of value); due to this optimization, it was possible to reduce memory costs by 10-15%. It is worth noting that the results obtained have not been tested on a "mass consumer"; in this case, the percentage of interested users (as opposed to the expert group of testing participants) will be predictably smaller and the gains from this optimization are expected to be significantly greater.

VII. FUTURE WORKS

In the following studies, it is planned to compare how the boundary of "specialization" of the correspondence context changes depending on the parameters of the model itself, and to experimentally test the "dreams" and "summarization" methods.

VIII. REFERENCES

[1] Hugo Touvron et al., Llama 2: Open Foundation and Fine-Tuned Chat Models. (Jul 18, 2023) DOI: 10.48550/arXiv.2307.09288

[2] Albert Q. Jiang et al., Mistral 7B. (Oct 10, 2023) DOI: 10.48550/arXiv.2310.06825

[3] OpenAI GPT-4 Technical Report (Mar 15 2023) DOI: 10.48550/arXiv.2303.08774

[4] Vaswani, Ashish, et al. Attention is all you need, Advances in neural information processing systems 30 (2017).

[5] Haoyi Xiong, Jiang Bian, Sijia Yang, Xiaofei Zhang, Linghe Kong, Daqing Zhang Natural Language based Context Modeling and Reasoning with LLMs: A Tutorial (Sep, 2023) DOI: 10.48550/arXiv.2309.15074

[6] C. Fang, A. Zhou and Z. Wang, "An Algorithm–Hardware Co-Optimized Framework for Accelerating N:M Sparse Transformers," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 11, pp. 1573-1586, Nov. 2022, doi: 10.1109/TVLSI.2022.3197282.

[7] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen and M. Chen, "In-Edge AI: Intelligentizing Mobile Edge Computing, Caching and Communication by Federated Learning," in IEEE Network, vol. 33, no. 5, pp. 156-165, Sept.-Oct. 2019, doi: 10.1109/MNET.2019.1800286.

[8] Kwon, Woosuk & Zhuohan, Li & Zhuang, Siyuan & Sheng, Ying & Zheng, Lianmin & Yu, Hao & Gonzalez, Joseph & Zhang, Hao & Stoica, Ion. (2023). Efficient Memory Management for Large Language Model Serving with PagedAttention. 611-626. 10.1145/3600006.3613165.

[9] B. Keller et al., "A 95.6-TOPS/W Deep Learning Inference Accelerator With Per-Vector Scaled 4-bit Quantization in 5 nm," in IEEE Journal of Solid-State Circuits, vol. 58, no. 4, pp. 1129-1141, April 2023, doi: 10.1109/JSSC.2023.3234893.

[10] T. L. Saaty, "Decision making for leaders," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 3, pp. 450-452, May-June 1985, doi: 10.1109/TSMC.1985.6313384.