On the Dependable Operation of Key-Value Caches in Large Language Models (LLMs)

Zhen Gao[1], Jie Deng[2], Shanshan Liu[3], Pedro Reviriego [4] and Fabrizio Lombardi[5]
[1] School of Electrical and Information Engineering, Tianjin University, Tianjin, 300072, China
[2] School of Future Technology, Tianjin University, Tianjin, 300072, China
[3] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China
[4] ETSI de Telecomunicación, Universidad Politécnica de Madrid, Madrid 28040, Spain
[5] Department of Electrical and Computer Engineering, Northeastern University Boston, MA 02215, USA
zgao@tju.edu.cn, dengjie2023@tju.edu.cn, pedro.reviriego@upm.es, ssliu@uestc.cn, lombardi@eoe.neu.edu

***Abstract*—The use of Transformer-based architectures has triggered the fast development of large language models (LLMs); LLMs achieve unprecedented performance for a wide range of natural language processing tasks. The Attention mechanism in LLMs is computationally intensive, so most LLMs choose to cache the Keys and Values vectors of existing tokens to achieve a trade-off between computational complexity and additional memory; this technique is generally known as KV cache. The size of this cache can be even larger than the memory storing the parameters, so, for its dependable operation as requirement in many applications, it is very important to assess its performance in the presence of soft errors in the memory. To the best of the authors' knowledge the impact of soft errors on the memory for the KV cache has not been previously studied. In this paper, the impact of bit-flip memory errors on the KV caches (with half-precision floating-point values) of two widely used LLMs (Mistral-7B and LlaMA2-7B) is evaluated based on error injection simulation. The results show that the first two exponent bits of the cache values are critical for LLM dependability, and errors on the prefilling stage tend to have a more severe impact than those on the decoding stage.**

***Index Terms*—Large Language Models (LLMs), Attention, Dependability, Key-Value cache, Soft error.**

I. Introduction

Transformers have a central role in machine learning (ML), enabling the development of large language models (LLMs) based tools such as ChatGPT [1]. An important innovation introduced by Transformers is Attention, a mechanism that enables capturing complex relations among words that may be distant in the provided text [2]. The Attention mechanism uses a triplet of Key, Value and Query to compute the weighted sum of the values; the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. This process incurs a significant computational effort.

To reduce the cost of implementing Attention, a key observation is that as each new token is predicted using the input prompt and previously generated tokens, most of the Attention Keys and Values are the same as in the previous tokens. Therefore, Keys and Values can be cached and reused for the next tokens. The use of a KV cache reduces the computational cost of implementing Transformers, but it increases memory use [3]. Depending on the length of the text, the memory needed by the KV cache can be similar, or even larger than the one needed to store the LLM parameters.

Dependability is one of key requirements for ML systems [4]; for LLMs, the impact of radiation-based upsets such as soft errors on the model parameters has been studied; for example for Bidirectional Encoder Representations from Transformers (BERT) it has been showed that there are critical bits on which errors can significantly degrade the model performance [5]. Since the KV cache can account for a significant part of the memory used by an LLM, it is important to understand the impact of soft errors on this cache; however, to the best of the authors' knowledge, this has not been studied.

In this paper, the dependability of LLMs when affected by errors on the KV cache is analyzed. Two popular open LLMs (Mistral-7B [6] and LlaMA2-7B [7]) are used as case studies for the evaluation. Section II covers the preliminaries, so introducing the two LLMs, the KV cache and the error model. The impact of errors on the KV cache is analyzed based on error injection experiments in Section III and the paper ends with the conclusion in Section IV.

II. Preliminaries

A. LLaMA2-7B and Mistral-7B

Like most LLMs, both LLaMA2 and Mistral models use a decoder-only Transformer architecture. As shown in Fig. 1, the LLaMA2 model includes three parts: an input embedding layer, a stack of N Transformer blocks and an output Linear layer [7]. For an input with M words, the embedding layer transforms it to a $W \times M$ matrix X, where each word is represented as a vector with an embedding dimension of W. Each decoder block is composed of the Self Attention (SA) and a Feed Forward Network (FFN). The SA part includes H heads, and each head calculates the $d_h \times M$ Query, Key and Value matrix based on linear processing as $Q = W_Q X + B_Q$, $K = W_K X + B_K$ and $V = W_V X + B_V$, where d_h is the head dimension, W_Q, W_K, and W_V are the weight matrices with dimension $d_h \times W$, and B_Q, B_K, and B_V are the bias vectors with dimension $d_h \times 1$. Then the SA is calculated as

$$SA(Q,K,V) = V \times softmax\left(K^T Q / \sqrt{d_h}\right). \quad (1)$$

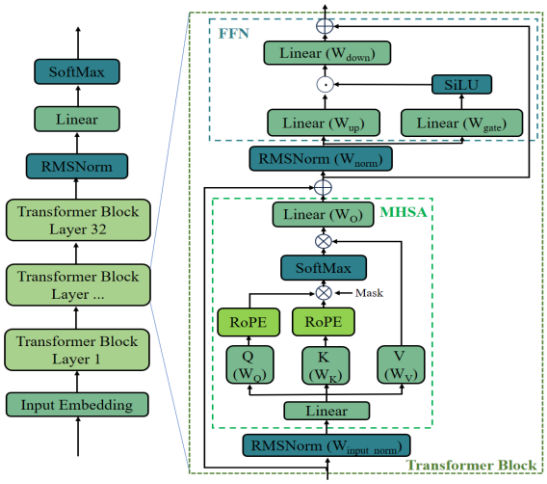


Fig. 1. Common Flow Diagram for LLaMA2 and Mistral

Let the result for the i^{th} head be denoted as $Head_i(X)$; the multiple heads are then concatenated and fused by linear processing as $W_O \times \text{Concat}(Head_i(X)) + B_O$, resulting in a $W \times M$ matrix denoted as MH-SA(X). After residual addition with the input X and normalization, the input to the FFN is expressed as $Y = norm(X + \text{MH-SA}(X))$.

The FFN is composed of three linear processing operations and a non-linear activation operation (SiLU) between them, so processing of the FFN is given by

$$\begin{cases} Y_{Gate} = SiLU(W_{Gate}Y + B_{Gate}) \\ Y_{Up} = W_{Up}Y + B_{Up} \\ Y_{Down} = W_{Down}(Y_{Up} \odot Y_{Gate}) + B_{Down} \end{cases}, \quad (2)$$

where W_{Gate}, W_{Up} and W_{Down} are the weight matrices for the three linear projections with dimension of $d_F \times W$ (d_F is the hidden dimension); B_{Gate}, B_{Up} and B_{Down} are the corresponding bias vectors, and $\odot$ denotes the dot project. After residual addition and normalization, the output of the encoder is given by $Z = norm(Y + Y_{Down})$, which is also a $W \times M$ matrix, so it can be fed as input to the next decoder layer. Finally, the output of the last decoder layer is fed to a linear layer to predict one output token. This is appended at the end of the input sequence to form a $W*(M+1)$ embedding matrix as new input for the prediction of the next output token.

With nearly same flow diagram, Mistral-7B makes several improvements [6]. For example, MH-SA is replaced by the Grouped Query Attention (GQA) [8], which shares the same Key and Value matrices for multiple Query matrices in a group; the Sliding Window Attention (SWA) technique is applied to limit the range of context for Attention calculation [9].

The main parameters for LLaMA2-7B and Mistral-7B are shown in Table I. Based on these parameters, it can be seen that the total number of weights in LLaMA2-7B or Mistral-7B is about 7 billion (as also suggested by their names).

Table I. Main parameters of LLaMA2-7B and Mistral-7B

Param.	Embed Dim (W)	# of layers (N)	# of Q heads (H)	# of KV heads	Head Dim (d_h)	Hidden Dim (d_F)
LLaMA2-7B	4096	32	32	32	128	11008
Mistral-7B	4096	32	32	8	128	14336

B. KV Cache

Based on the previous description, LLMs are auto-regressive models, i.e., the next output token is predicted based on the input prompt and previously generated tokens. If the input prompt includes s tokens and there are t tokens generated, then the first $s+t$ rows of K and V matrices must be re-computed for the (t+1)-th output token, which is a waste of computation resource. In practice, fast response latency is a key objective for service providers. Therefore, most LLMs choose to cache the K and V vectors of existing tokens to achieve a trade-off between computational complexity and additional memory; this technique is generally known as KV cache [3].

When the KV cache is utilized, the workload of LLMs is usually divided into two stages: *prefilling* and *decoding*. In the prefilling stage, all M vectors in the K and V matrix for the input prompt are calculated together for all heads in all layers. In the decoding stage, a new row for the K and V matrix are generated for the current inference (all heads in all layers), this is appended to the cached K and V matrix for the next inference.

The KV cache values are often stored by default as half-precision floating-point values (FP16). For an input prompt with s tokens and a generated output with t tokens, the memory for the all cached K and V matrices (in bytes) can be calculated as $B*N*H_{KV}*(s+t)*2*2$, where B, N and H_{KV} are the batch size, the number of layers and the number of KV heads in the LLM, respectively (first '2' is for the K cache and the V cache, and second '2' is for the FP16 format). The largest values of (s + t) for LLaMa2-7B and Mistral7B are 4096 and 8192, respectively. Therefore, for a normal batch size, e.g., 128, the memories for the KV cache can be up to 256 GB and 128 GB for LLaMA2-7B and Mistral-7B; this is significantly larger than the memory needed for the parameters (around 14 GB for the FP16 format).

C. Error Model

One of the major dependability concerns for computing systems is the presence of soft errors on memories; soft errors may be caused by radiation-effects [10], [11], voltage scaling for power saving [12] or changes in physical properties (such as the resistance drift in emerging memristor [13] or phase change memories [14]). Soft errors can flip a memory bit changing its value; depending on the value stored on the bit, this may degrade the operation of the system. In this paper, we focus on soft errors in memories used to store the KV cache. Three typical soft error rates (SER) that are often used in related works [15], [16]are considered, i.e., 10^{-4}, 10^{-3} and 10^{-2}.

III. Dependability Evaluation

A. Simulation Setup

The Massive Multitask Language Understanding (MMLU) is a widely used benchmark to test the ability of LLMs to answer questions on different topics [17]. It has 15,908 questions on 57 categories, and we select the 170 questions from the "world religions" category for evaluation in this paper. All questions are formulated as multiple-choice questions, and the LLM provides an answer as A, B, C or D

based on the generated output text. Then, the result is compared with the correct answer, and the accuracy is calculated as the ratio of the correct answers over the total number of questions.

With the implementations of [6] and [7], the accuracy of Mistral-7B and LLaMA2-7B on the "world religions" category of MMLU in the error-free case is 63.5% and 45.3%, respectively. The accuracy loss with errors on the KV cache is used in this paper as a metric for the impact of errors on LLM performance. In particular, four experiments are designed for the dependability analysis: 1) random errors on the KV cache for all heads in all layers to get a general understanding of the impact of errors on LLM performance; 2) errors on specific bit positions to identify the critical bits for LLM dependability; 3) errors on a specific layer to check the error sensitivity for different layers; 4) different portions of errors on prefilling and decoding stages to identify the stage that is more prone to errors on the KV cache. All experiments are performed on both LLM models (Mistral-7B and LLaMA2-B), so that we can check if the findings are model specific.

B. General Impact of Errors on the KV Cache

In this experiment, we first select some values from the K or V cache for each layer based on a specific SER, and then randomly flip one of the 16 bits of each selected FP16 value. With the injected bit errors, we run the LLM on the test set and measure the accuracy; such experiment is repeated 50 times to get the average accuracy value. The results for the two LLMs are shown in Table II; errors with an SER of 10^{-4} significantly degrade LLM performance (< 10%), and a higher SER causes the LLM to fail (0 accuracy).

Table II. Task Performance with errors on KV cache

	K cache			V cache		
SER	10^{-4}	10^{-3}	10^{-2}	10^{-4}	10^{-3}	10^{-2}
Mistral-7B	6.16%	0	0	6.74%	0	0
LLaMA2-7B	4.73%	0	0	5.87%	0	0

C. Impact of Errors on Different Bit Positions

In this experiment, after selecting some values from the K or V cache for each layer, we only flip a specific bit (e.g., bit-*x*) of each selected FP16 value, and then test the accuracy with those errors. The experiment is repeated 10 times to get the average accuracy value. The results for Mistral-7B and LLaMA2-7B are shown in Fig. 2 and Fig. 3, respectively, where bit-1 refers to the sign bit, bit-2 to bit-6 refer to the exponent bits and the others are fractional bits. Based on Fig. 2, three observations are possible: 1) errors on the 1st exponent bit cause the LLM to totally fail even for an SER of 10^{-4}; 2) errors on the first 2 or 3 exponent bits cause an obvious accuracy loss, while those on other bits do not affect the LLM performance; 3) the accuracy loss by errors on the K cache is significantly larger than for errors on the V cache. Similar conclusions can be found in Fig. 3 for LLaMA2-7B, but there are two main differences. The first difference is that the accuracy loss by errors on the 2nd exponent bit is very small even for an SER of 10^{-2}. The other is that the difference between the K and V cache is also very small.

These conclusions can be explained by the amplitude of the K V cache. As shown in Fig. 4, the amplitude of the K cache is larger than the V cache for Mistral-7B, and most values are smaller than 2. In this range, the 1st (2nd) exponent bit is likely to be 0 (1) at a high probability (about 92% in both cases). Therefore, most errors on the first 1st exponent bit cause 0→1 flips, which increase the value by 2^{16}. Such large changes on some values of K or V cache can spread to all items in the layer output, and finally generate unreadable output text. For the 2nd exponent bit, most errors cause 1→0 flips, and only a small portion of the errors increases the value by 2^{8}, so the impact is rather smaller. Errors on other bits have a smaller impact due to the small changes in values. Furthermore, as shown in Fig. 5, the amplitude of the K cache for LLaMA2-7B is smaller than for Mistral-7B. In this case, the 2nd exponent bit is likely to be 1 at a probability of 98% for the K cache, so only 2% of the values can cause large errors and thus, introducing small differences for the impact of errors on the K and V caches.

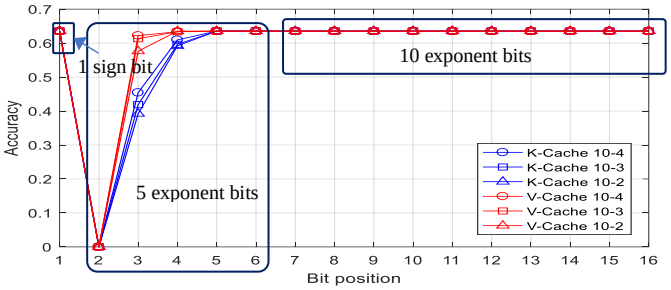


Fig. 2. Effect of errors on different bits in all layers (Mistral-7B)

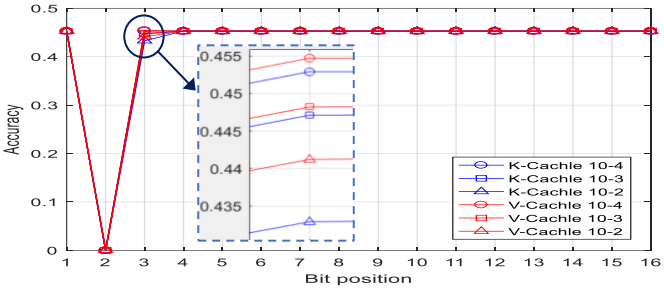


Fig. 3. Effect of errors on different bits in all layers (LLaMA2-7B)

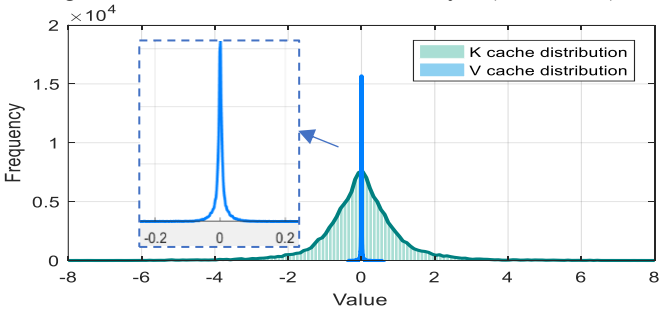


Fig. 4. Amplitude distribution for KV cache (Mistral-7B)

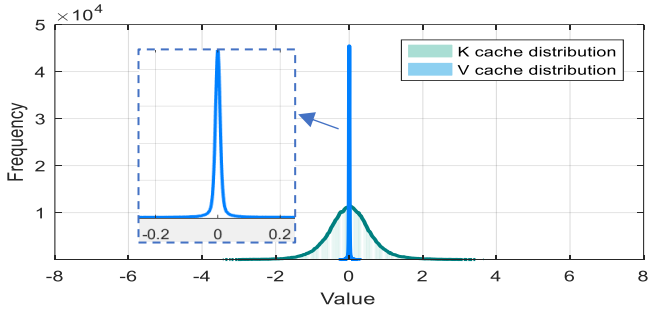


Fig. 5. Amplitude distribution for KV cache (LLaMA2-7B)

D. *Impact of Errors on Different Layers*

To assess the impact of errors on different layers, we only inject errors on the 2nd exponent bit for some values in a specific layer; the results for each layer are then averaged over 10 runs. Fig. 6 provides the results for Mistral-7B with an SER of 10^{-2}. The impact of errors on different layers is nearly the same. Results for other SERs and for LLaMA2-7B support the same conclusion, hence different layers have the same vulnerability to errors on the KV cache.

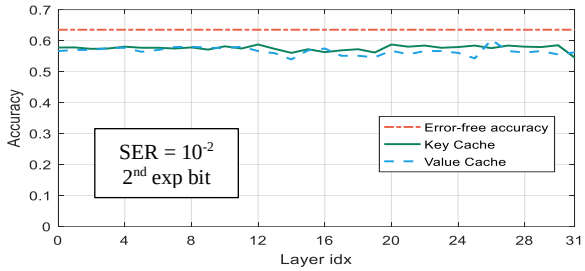
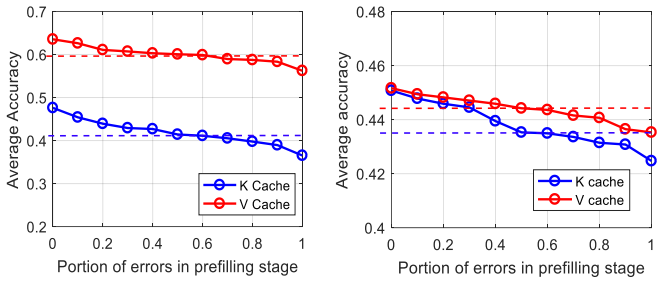


Fig. 6. Effect of errors on KV cache for different layers (Mistral-7B)

E. *Impact of Errors on Two Stages*

In this experiment, we inject errors on the KV cache in all layers, but only flip the 2nd exponent bit of each selected value. We control the portion of errors on the prefilling stage (p) and the decoding stage (d), where p and d are values between 0 and 1 and $p + d = 1$. For example, if we inject all errors for a specific SER into the KV cache for the input prompt, the portion of errors on the prefilling stage is 1 and on the decoding stage it is 0. Then we gradually change the value of p from 0 to 1 and examine the impact of the same number of errors on task performance. The results for Mistral-7B and LLaMA2-7B are shown in Fig. 7, the dashed line is the accuracy for equal portion on two stages ($p = d = 0.5$). Each point in the figures is averaged over 10 runs. The two figures give the same conclusion, i.e., a higher portion of errors on the prefilling stage tends to cause a larger performance degradation. This implies that the input prompt part is more prone to errors than the generated text; this is reasonable because the LLM is less likely to give correct answer if the input prompt is harder to understand.



(a) Mistral-7B (b) LLaMA2-7B

Fig. 7. Impact of errors for different portion on two stages

I. Conclusion

LLMs achieve very good performance in many tasks, but their dependable operation is challenging in the presence of soft errors in the memories. The KV cache is a commonly applied technique in LLMs to speed the inference by avoiding re-computation; however, the memory for the KV cache can be similar or even larger than for the parameters. This paper evaluates the dependability of LLMs in the presence of soft errors on the KV cache for the first time. Error injection for two popular LLMs (Mistral-7B and LLaMA2-7B) have revealed the following general conclusions: 1) errors on the KV cache degrade the LLM performance dramatically, those on the first two exponent bits account for most of the degradation; 2) errors on different layers have roughly the same impact on the LLM performance; 3) errors on the KV cache in the prefilling stage tend to cause a larger accuracy loss than errors on the decoding stage. These results will be useful to design protection schemes for the KV cache against soft errors in memories.

References

[1] T. Wu, S. He, J. Liu, et al., “A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development,” IEEE/CAA Journal of Automatica Sinica, vol. 10, no. 5, May 2023.

[2] A. Vaswani et al. “Attention is all you need”, in proceedings of Advances in Neural Information Processing Systems, 2017.

[3] S. Luohe, et al. "Keep the Cost Down: A Review on Methods to Optimize LLM's KV-Cache Consumption." Conference On Language Modeling 2024, Philadelphia, PA, Oct. 7-9, 2024.

[4] F. Su, C. Liu and H. -G. Stratigopoulos, "Testability and Dependability of AI Hardware: Survey, Trends, Challenges, and Perspectives," IEEE Design & Test, vol. 40, no. 2, April 2023.

[5] Z. Gao, et al. On the Dependability of Bidirectional Encoder Representations from Transformers (BERT) to Soft Errors, IEEE Transactions on Nanotechnology, vol. 24, pp. 73-87, Jan. 2025.

[6] A. Q. Jiang et al, “Mistral 7B”, ArXiv e-prints, 2023. doi:10.48550/arXiv.2310.06825.

[7] H. Touvron et al. "Llama 2: Open foundation and fine-tuned chat models." arXiv preprint arXiv:2307.09288, 2023.

[8] J. Ainslie, J. L. Thorp, M. D. Jong, et al., “GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints,” EMNLP 2023, Singapore, Dec. 2023.

[9] I. Beltagy, et al., “Longformer: The Long-Document Transformer,” arXiv preprint arXiv:2004.05150, Dec. 2020.

[10] R. C. Baumann, “Radiation-induced Soft Errors in Advanced Semiconductor Technologies,” IEEE Transactions on Device and materials reliability, vol. 5, no. 3, pp. 305-316, 2005.

[11] C. Bolchini, L. Cassano, A. Miele, et al., "Fast and Accurate Error Simulation for CNNs Against Soft Errors," in IEEE Transactions on Computers, vol. 72, no. 4, pp. 984-997, 1 April 2023.

[12] F. Frustaci, et al., “SRAM for error-tolerant applications with dynamic energy-quality management in 28 nm CMOS,” IEEE Journal of Solid-state circuits, vol. 50, no. 5, pp. 1310–1323, 2015.

[13] Y. Yilmaz and P. Mazumder, “A Drift-tolerant Read/write Scheme for Multilevel Memristor Memory,” IEEE Transactions on Nanotechnology, vol. 16, no. 6, pp. 1016-1027, 2017.

[14] P. Junsangsri. and F. Lombardi, "A New Comprehensive Model of a Phase Change Memory (PCM) Cell", IEEE Transactions on Nanotechnology, vol 13, no. 6, pp. 1213-1225, 2014.

[15] E. Ozen and A. Orailoglu, “Sanity-check: Boosting the Reliability of Safety-critical Deep Neural Network Applications,” in IEEE 28th Asian Test Symposium (ATS), 2019.

[16] Z. Gao, et al., “Reducing the Energy Dissipation of Large Language Models (LLMs) with Approximate Memories”, International Symposium on Circuits and Systems (ISCAS), 2024.

[17] D. Hendrycks, C. Burns, S. Basart, et al., “Measuring Massive Multitask Language Understanding,” ICLR 2021.