VIII Simpósio de Iniciação Científica 2025

"Do conhecimento acadêmico à transformação sustentável: inovação com validação científica"

ESTRATÉGIAS DE GERENCIAMENTO DE KV CACHE: UM ESTUDO COMPARATIVO DE DESEMPENHO E MEMÓRIA

Mirela V. Domiciano[1] (EG), Ryan F. F. Ribeiro[1] (EG), Egon L. Muller Junior (PQ)[1]
[1]*Universidade Federal de Itajubá - Campus Itajubá.*

Palavras-chave: Desempenho de inferência. KV Cache. Modelos de Linguagem de Grande Escala. Otimização de memória.

Introdução

Modelos de linguagem de grande escala (LLMs), baseados predominantemente na arquitetura Transformer, tornaram-se a fundação de inúmeras aplicações de inteligência artificial. A geração de texto nesses modelos ocorre de forma autorregressiva, onde a predição de cada novo token depende das representações internas dos tokens anteriores. Para otimizar este processo e evitar a recomputação onerosa das representações de atenção (chaves e valores) a cada passo, utiliza-se uma técnica de cache denominada KV Cache (POPE et al., 2023). Embora acelere significativamente a inferência, o KV Cache consome uma quantidade substancial de memória, especialmente em contextos longos, tornando-se um dos principais gargalos para a implementação eficiente de LLMs (LIU et al., 2024). A gestão eficaz deste cache é, portanto, crucial para o avanço e a viabilidade de modelos com janelas de contexto cada vez maiores.

Diante desse cenário, o objetivo central desta pesquisa é conduzir um estudo comparativo de estratégias fundamentais de gerenciamento de KV Cache, avaliando sistematicamente seu impacto no desempenho e no consumo de memória durante a inferência de LLMs. Serão investigadas e comparadas três abordagens essenciais: a ausência de cache, que serve como linha de base para mensurar o custo computacional bruto; o cache dinâmico, implementação padrão na maioria dos frameworks que aloca memória sob demanda; e o cache estático, uma alternativa que pré-aloca um espaço fixo visando otimizar a velocidade de acesso.

A justificativa para este estudo reside na crescente demanda por LLMs capazes de processar documentos extensos, manter diálogos longos e realizar tarefas complexas que exigem um vasto contexto. O gerenciamento inadequado do KV Cache não apenas limita o comprimento máximo da sequência que um modelo pode suportar, mas também degrada a vazão (*throughput*) e aumenta a latência, impactando diretamente a experiência do usuário e os custos operacionais associados à infraestrutura de hardware. A otimização do KV Cache é, portanto, um passo fundamental para democratizar o acesso a LLMs de ponta, permitindo sua execução em hardware com recursos mais limitados e a redução da latência de resposta. Este trabalho busca, assim, fornecer uma análise clara dos trade-offs entre compressão de memória e performance, para embasar o desenvolvimento e a implementação de sistemas de IA mais eficientes e escaláveis.

Metodologia

O estudo foi conduzido utilizando a plataforma Google Colab, com acesso a um ambiente de execução com GPU (Graphics Processing Unit) para acelerar os processos de inferência do modelo de linguagem. A implementação foi desenvolvida integralmente na linguagem de programação Python, com o suporte de bibliotecas essenciais para aprendizado de máquina e manipulação de dados.

Para a interação com o modelo de linguagem, utilizou-se o framework Transformers da Hugging Face, que fornece uma interface de alto nível para carregar e executar modelos de última geração. A biblioteca PyTorch foi empregada como backend para todas as operações de tensores e gerenciamento de memória na GPU. Para a mensuração de métricas de desempenho sistêmico, como o uso de memória, foram utilizadas as bibliotecas psutil e as funções nativas de alocação de memória do PyTorch.

O modelo de linguagem de grande escala (LLM) selecionado para os testes foi o meta-llama/Llama-3.2-1B. A escolha deste modelo se justifica por ser uma arquitetura Transformer

representativa, de código aberto e com um tamanho que permite a execução de múltiplos testes em um ambiente com recursos limitados, como o Google Colab, sem sacrificar a relevância dos resultados. O modelo e seu respectivo tokenizador foram carregados diretamente do repositório da Hugging Face

Para avaliar as estratégias de cache em um contexto prático e realista, foi desenvolvido um benchmark conversacional. Este benchmark simula interações de múltiplos turnos entre um usuário e um agente de inteligência artificial em diferentes domínios, como atendimento de seguros, suporte bancário e e-commerce. Cada cenário é composto por:

- Prompt de Sistema: Uma instrução inicial que define a identidade, o papel e as regras de comportamento do agente de IA.
- Sequência de Turnos: Uma lista predefinida de perguntas de um usuário, projetada para aumentar progressivamente o contexto da conversa.

Essa abordagem permite analisar o comportamento do KV Cache à medida que o histórico da conversa cresce, simulando uma aplicação real onde o contexto precisa ser mantido eficientemente ao longo do tempo.

O núcleo da pesquisa consistiu na comparação de três estratégias distintas de gerenciamento de cache, implementadas através da biblioteca Transformers:

- Sem Cache: Uma linha de base onde o mecanismo de cache é desativado. Nesta configuração, os estados de atenção (chaves e valores) são recalculados do zero para toda a sequência a cada novo token gerado.
- Cache Dinâmico: A implementação padrão e mais comum. O cache cresce dinamicamente para acomodar novos tokens a cada etapa de geração. É flexível, mas pode incorrer em sobrecarga de alocação de memória.
- Cache Estático: Uma abordagem onde o cache é pré-alocado com um tamanho máximo fixo. Isso pode reduzir a sobrecarga de alocação dinâmica, potencialmente oferecendo maior velocidade em detrimento de uma maior alocação inicial de memória.

O experimento foi executado de forma automatizada. Para cada um dos cenários conversacionais, o sistema itera sobre as três estratégias de cache. Em cada turno da conversa, o histórico era atualizado com a pergunta do usuário, e o modelo era invocado para gerar uma resposta. Durante este processo, as seguintes métricas foram coletadas:

- Tempo de Geração (segundos): O tempo total decorrido para a geração da resposta do modelo em cada turno.
- Uso de Memória (MB): O aumento na alocação de memória da GPU atribuível à operação de geração, medido como a diferença entre a memória alocada antes e depois da chamada ao método generate.
- Vazão (tokens por segundo): A velocidade de geração, calculada como o número de tokens gerados dividido pelo tempo de geração.

Ao final da execução de cada cenário, os resultados detalhados de cada turno e estratégia foram salvos em um arquivo no formato CSV análise posterior, garantindo a rastreabilidade e reprodutibilidade dos dados coletados.

Resultados e discussão

Nesta seção, apresentam-se os principais resultados obtidos durante a pesquisa. Para validar e comparar as diferentes estratégias de gerenciamento de KV Cache, foi executado um benchmark conversacional simulando múltiplos turnos de diálogo. O modelo meta-llama/Llama-3.2-1B foi submetido a cenários de conversação que aumentavam progressivamente o contexto. Foram avaliadas três configurações: sem cache (*None*), com cache dinâmico (*Dynamic*) e com cache estático (*Static*). As principais métricas de desempenho coletadas foram: tempo de geração, uso de memória e vazão (tokens/s).

Os resultados agregados, apresentando a média de desempenho de cada estratégia ao longo de 90 amostras de conversação, estão consolidados na Tabela 1.

Estratégia	Tempo Geração (s)	Uso Memória (MB)	Tokens/s
None	54,2749	0,009814	6,3264
Dynamic	3,4433	0,009814	34,1041
Static	4,8570	4,51370	34,4357

Tabela 1 – Resumo dos Indicadores de Desempenho por Estratégia de Cache

A análise do tempo de geração revela o impacto crucial do uso de cache na inferência de LLMs. Como ilustrado na Figura 1, a estratégia *None* (sem cache) demonstrou um aumento exponencial no tempo de resposta à medida que o contexto da conversa (número de turnos) crescia. O tempo médio de 54,27 segundos torna essa abordagem inviável para aplicações interativas. Em contrapartida, as estratégias *Dynamic* e *Static* mantiveram um tempo de geração baixo e estável, com médias de 3,44 e 4,86 segundos, respectivamente, evidenciando uma melhoria de desempenho superior a 10 vezes em comparação com a ausência de cache.

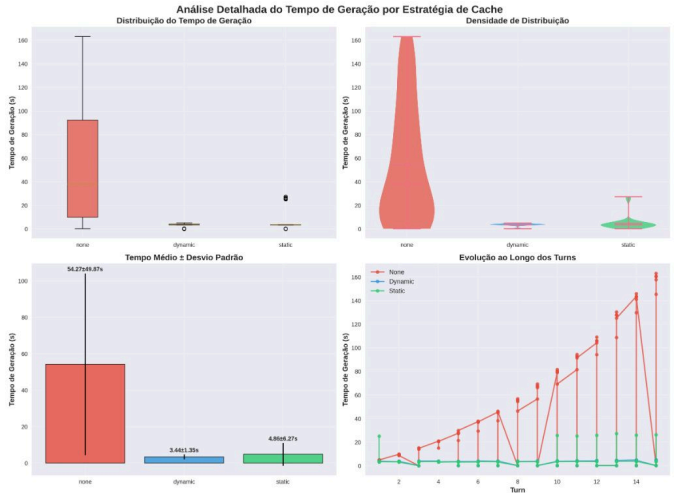

Figura 1 – Análise Detalhada do Tempo de Geração por Estratégia de Cache

A vazão, medida em tokens por segundo, corrobora esses achados. A Figura 2 mostra que as estratégias com cache *(Dynamic* e *Static*) alcançaram uma média de aproximadamente 34 tokens/s, enquanto a abordagem sem cache ficou limitada a apenas 6,3 tokens/s. O gráfico de dispersão também indica que, embora a vazão de todas as estratégias diminua com o aumento do contexto, a queda é drasticamente mais acentuada na ausência de cache.

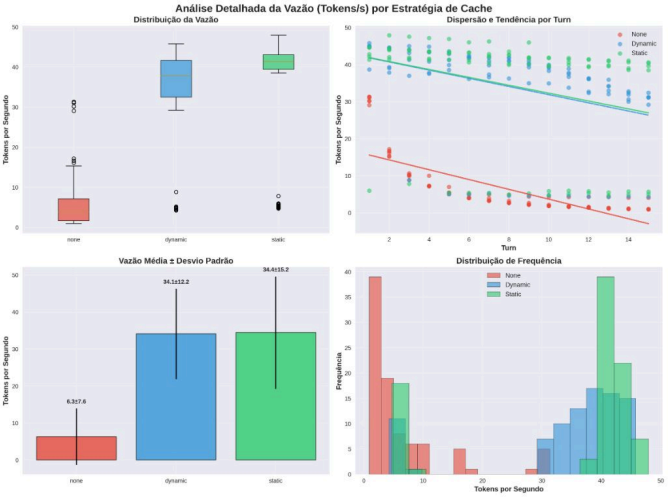

Figura 2 – Análise Detalhada da Vazão por Estratégia de Cache

O uso de memória apresentou o principal trade-off entre as estratégias de cache. A Figura 3 detalha o consumo de memória adicional durante cada etapa de geração. As estratégias *None* e *Dynamic* registraram um consumo marginal e constante por turno (aproximadamente 0.01 MB), pois a medição captura apenas o delta de alocação durante a inferência. No entanto, a estratégia *Static* demonstrou um comportamento distinto, com um consumo médio significativamente maior, de 4,51 MB.

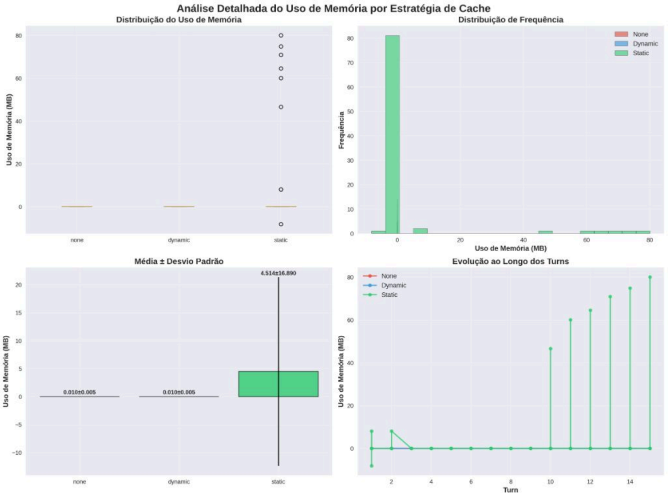

Figura 3 – Análise Detalhada do Uso de Memória por Estratégia de Cache.

Em suma, os resultados confirmam que o gerenciamento de KV Cache é indispensável para o desempenho de LLMs em tarefas conversacionais. A estratégia *Dynamic* foi a mais equilibrada, oferecendo uma redução no

tempo de geração com um impacto mínimo no consumo de memória por inferência. A estratégia *Static*, embora igualmente rápida, impõe uma penalidade de memória considerável, tornando-a mais adequada para cenários onde o comprimento máximo da sequência é conhecido e a memória pode ser provisionada com antecedência. A ausência de cache, mostrou-se significativamente mais lenta.

Conclusões

Este estudo demonstrou de forma conclusiva a importância crítica do gerenciamento de KV Cache para a viabilidade e o desempenho de Modelos de Linguagem de Grande Escala (LLMs) em aplicações conversacionais. A pesquisa comparou sistematicamente três estratégias de cache — sem cache, dinâmico e estático — e quantificou o impacto de cada uma no tempo de geração, no consumo de memória e na vazão de tokens.

Os resultados obtidos evidenciam que a ausência de um mecanismo de cache faz com que o tempo de resposta aumente exponencialmente à medida que o contexto da conversa se expande. Em contrapartida, o uso de cache, tanto dinâmico quanto estático, resultou em uma melhoria de desempenho superior a 10 vezes, mantendo o tempo de geração baixo e estável. A estratégia de cache dinâmico destacou-se como a mais equilibrada, alcançando uma alta vazão de aproximadamente 34 tokens por segundo com um consumo de memória adicional mínimo por inferência. Embora a estratégia de cache estático tenha apresentado uma velocidade similar, seu custo em termos de alocação de memória foi consideravelmente maior e mais volátil, tornando-a menos ideal para ambientes com recursos de memória limitados ou imprevisíveis.

Conclui-se, portanto, que a estratégia de cache dinâmico representa a solução mais robusta e eficiente para a maioria das aplicações de LLMs, oferecendo um excelente equilíbrio entre velocidade e uso de recursos. As descobertas desta pesquisa servem como uma base sólida para o desenvolvimento de sistemas de IA mais otimizados. Para trabalhos futuros, sugere-se a expansão deste estudo para incluir a análise de técnicas mais avançadas, como políticas de evicção de tokens (por exemplo, H2O) e métodos de quantização (como KVQuant), que prometem otimizar ainda mais o uso de memória, permitindo que os LLMs processem contextos ainda mais longos de forma eficiente.

Agradecimentos

Agradecemos aos colegas e professores do PET-TEC por promoverem um ambiente de inovação tecnológica que permitiu o desenvolvimento deste trabalho. Nossa gratidão também à UNIFEI e ao FNDE pelo suporte e incentivo, essenciais para a realização deste projeto.

Referências

LIU, S., et al. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. arXiv preprint arXiv:2405.09980, 2024. Disponível em: https://arxiv.org/abs/2405.09980. Acesso em: ago. 2025.

POPE, R., et al. Efficiently Scaling Transformer Inference. In: Proceedings of the 6th MLSys Conference, 2023. Disponível em: https://arxiv.org/abs/2211.05102. Acesso em: ago. 2025.

ZHANG, Z., et al. H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In: Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS), 2023. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf. Acesso em: ago. 2025.

GOOGLE. Colaboratory. Disponível em: https://colab.research.google.com/. Acesso em: ago. 2025.

HUGGING FACE. Transformers Documentation. Disponível em: https://huggingface.co/docs/transformers. Acesso em: ago. 2025.

PSUTIL. psutil Documentation. Disponível em: https://psutil.readthedocs.io/. Acesso em: ago. 2025.

PYTHON SOFTWARE FOUNDATION. Python Language Reference. Disponível em: https://www.python.org/. Acesso em: ago. 2025.

PYTORCH. PyTorch Documentation. Disponível em: https://pytorch.org/. Acesso em: ago. 2025.

META AI. The Llama 3 Suite of Models. Disponível em: https://ai.meta.com/blog/llama-3-1-405b/. Acesso em: ago. 2025.