Avaliação de Desempenho e Consumo de Energia da Classificação do Filtro de Loop Adaptativo do VVC em Múltiplos Paradigmas e Plataformas

VITÓRIA DE SOUZA FABRICIO[1]; IAGO STORCH[1]; GUILHERME CORREA[1], DANIEL PALOMINO[1]

[1]*Universidade Federal de Pelotas*
vitoria.sfabricio@gmail.com, {icstorch, gcorrea, dpalomino}@inf.ufpel.edu.br

1. INTRODUÇÃO

A crescente demanda por compressão de vídeo de alta eficiência impulsionou o desenvolvimento de novos padrões (Bossen, 2021), como o *Versatile Video Coding* (VVC), que integra ferramentas inovadoras para ganhos superiores de compressão (Bross 2021). Uma dessas ferramentas é o Filtro de Loop Adaptativo (ALF), que melhora significativamente a qualidade visual, mas ao custo de uma carga computacional elevada (Karczewicz, 2021). Esse alto custo não se reflete apenas em longos tempos de processamento, mas também em um consumo de energia significativo, uma preocupação crítica para dispositivos móveis e sistemas embarcados com recursos limitados (Saha, 2023).

O ALF opera classificando blocos de 4×4 da imagem com base em suas propriedades de textura para aplicar filtros espaciais que minimizam artefatos de codificação (Farhat, 2022). Embora trabalhos anteriores já tenham proposto otimizações (Fang 2023; Yin, 2023) e avaliado o tempo de processamento em diferentes paradigmas de programação em uma plataforma *desktop* (Saha, 2023; Fabricio, 2023), uma análise abrangente que inclua o consumo de energia e o desempenho em sistemas embarcados no codificador VTM ainda não foi realizada.

Este trabalho expande o estudo inicial (Fabricio, 2025), apresentando uma avaliação completa da etapa de classificação do ALF. A análise é estendida em duas dimensões principais: a primeira inclui o consumo de energia como uma métrica de avaliação, e a segunda compara o desempenho em duas plataformas de *hardware* distintas: um desktop de alto desempenho e um sistema embarcado de baixo consumo, o NVIDIA Jetson TX2. O objetivo é fornecer uma caracterização detalhada dos *trade-offs* entre velocidade de processamento e eficiência energética.

2. METODOLOGIA

2.1 Visão Geral do Filtro de Loop Adaptativo no VVC

O ALF do VVC melhora a qualidade da imagem ao corrigir alguns artefatos inseridos pelo processo de codificação. Ele atua após o quadro inteiro já ter sido codificado, ou seja, após as etapas de predição, transformação e quantização como observado na Figura 1. Além disso, o ALF é aplicado após dois outros filtros: o *Deblocking Filter* (DBF) e o *Sample Adaptive Offset* (SAO).

O processo do ALF começa na partição em blocos de 4×4 amostras. No entanto, para determinar a classe de cada bloco, uma região de 8×8 amostras centrada no bloco é considerada. Cada bloco é classificado em uma de 25 classes de acordo com a equação (1). Este cálculo é baseado da direção de textura D e na atividade de textura local quantizada Ã.

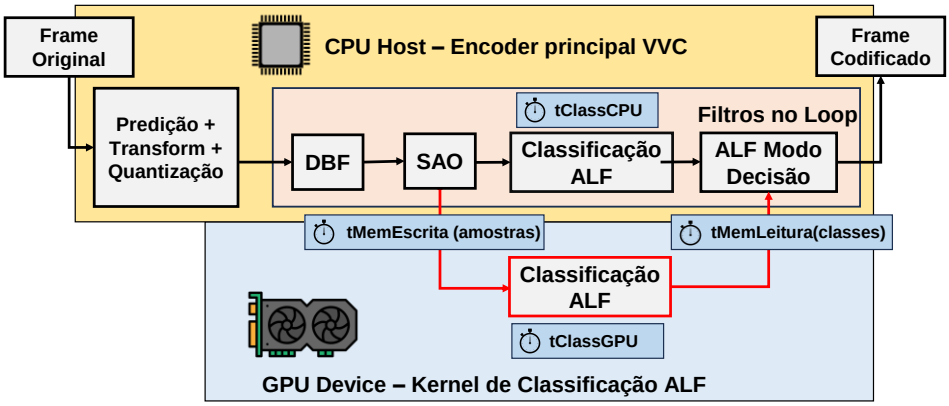

Figura 1. Visão geral do sistema de codificação CPU+GPU. Os contornos em vermelho destacam o desvio da GPU utilizado para a classificação ALF.

Para calcular D, são utilizados gradientes em quatro direções: horizontal, vertical e diagonais principal e secundária – representados por gh, gv, gd1 e gd2, respectivamente. Estes gradientes são obtidos usando o operador Laplaciano 1D discreto em uma janela de 8x8 amostras que envolve o bloco 4×4, além da vizinhança com duas amostras de espessura ao seu redor.. A direção D de cada bloco é definida a partir da relação dos valores de gradiente com um conjunto de limiares definidos pelo padrão VVC (Karczewicz, 2021).

Além disso, a atividade A é calculada a partir destes mesmos valores de gradientes verticais e horizontais, ajustando a profundidade de *bits* para obter a quantização Ã. Esse processo de classificação tem dependências de dados mínimas, permitindo a execução paralela em arquiteturas de computação modernas, o que é crucial para otimizar o desempenho do ALF em sistemas de codificação de vídeo.

$$C = 5C + A \qquad (1)$$

2.2 Implementações em CPU: Execução sequencial e paralelismo SIMD

O padrão VVC conta com um *software* codificador de referência, o codificador VTM (JVET, 2024), que implementa todas as ferramentas de codificação disponíveis no VVC, incluindo o ALF. Além disso, algumas ferramentas de codificação são implementadas com instruções escalares e SIMD para diferentes arquiteturas. Como essas implementações já estão disponíveis, este trabalho as considera para avaliar o desempenho da classificação ALF baseada em CPU. Na versão escalar ($VTM_{escalar}$), os blocos 4×4 são processados de maneira sequencial, e o cálculo do gradiente também é realizado de forma sequencial dentro de cada bloco.

Na versão SIMD (VTM_{SIMD}), o paralelismo é explorado através de instruções vetoriais, como *mm256_set1_epi32* e *mm_lddqu_si128*, que permitem o processamento simultâneo de várias amostras durante o cálculo do gradiente. Ainda assim, os blocos 4×4 são processados de forma sequencial.

2.3 Implementação em GPU: Programação massivamente paralela

Foi desenvolvida uma implementação em CUDA para executar a classificação ALF de forma massivamente paralela. Dois *kernels* foram projetados: o *computeGradient*, que calcula o gradiente de todas as amostras de uma região 32×32 do quadro em paralelo, e o *classifyBlock*, que classifica todos os blocos 4×4. A comunicação entre a CPU e a GPU ocorre transferindo as amostras reconstruídas para a GPU, onde o processamento é realizado e os resultados são devolvidos à CPU para a aplicação dos filtros. A arquitetura CUDA permite o uso

eficiente de milhares de *threads*, organizadas em *grids* de blocos, otimizando o uso de memória compartilhada e global para minimizar latências e maximizar a reutilização de dados.

2.4 Configuração Experimental

A avaliação foi conduzida em duas plataformas de *hardware* distintas para uma análise comparativa. A primeira foi um sistema *desktop* de alto desempenho, equipado com um processador Intel Core i7-8700 com o conjunto de instruções AVX2, 16 GiB de RAM e uma GPU dedicada NVIDIA RTX 3080. Em contrapartida, utilizou-se um sistema embarcado, a placa NVIDIA Jetson TX2, que possui uma CPU ARM Cortex-A57 e uma GPU integrada Pascal de 256 núcleos. É relevante notar que, embora o software de referência do VVC (VTM) possua implementações SIMD para diversas arquiteturas, uma otimização análoga para ARM não está disponível. Consequentemente, os testes executados na CPU da Jetson TX2 utilizaram apenas processamento escalar. Para os testes, foram processados os primeiros 32 quadros de sequências de vídeo com resoluções de 480p, 720p, 1080p e 4K, em conformidade com as Condições de Teste Comum (CTC) do VVC. No entanto, por limitações de capacidade de processamento, os testes em 1080p e 4K não puderam ser concluídos na CPU da Jetson TX2. A medição de desempenho focou no tempo de execução da etapa de classificação, incluindo a sobrecarga de comunicação entre CPU e GPU. O consumo de energia foi aferido com precisão utilizando ferramentas específicas para cada componente: RAPL para a CPU do desktop, a biblioteca NVML para a GPU dedicada e os monitores de energia integrados INA3221 para a placa Jetson TX2.

3. RESULTADOS E DISCUSSÃO

A Tabela 1 resume os resultados de tempo de execução e consumo de energia para cada plataforma e paradigma, utilizando vídeos 480p, 720p, 1080p, 4k como referência comparativa. Como mencionado anteriormente, a Jetson não é capaz de codificar vídeos 1080p e 4k em CPU por falta de recursos.

Tabela 1. Tempo [ms] e consumo de energia [J] médio por quadro.

Dispositivo	Métrica	480p	720p	1080p	4k
Desktop CPU Escalar Intel i7-8700 ($VTM_{escalar}$)	Tempo [ms]	1,600	3,950	8,950	35,97
	Energia [J]	0,032	0,064	0,138	0,523
Desktop CPU SIMD Intel i7-8700 (VTM_{SIMD})	Tempo [ms]	0,300	0,930	1,900	8,525
	Energia [J]	0,020	0,021	0,042	0,176
Desktop GPU NVIDIA RTX 3060	Tempo [ms]	1,305	3,244	6,512	27,40
	Energia [J]	0,043	0,106	0,051	0,899
	Comunicação	82%	84%	86%	89%
	Execução	18%	16%	14%	11%
Jetson TX2 CPU ARM Cortex A57	Tempo [ms]	5,300	12,00	N/A	N/A
	Energia [J]	0,006	0,013	N/A	N/A
Jetson TX2 GPU NVIDIA 256 cores	Tempo [ms]	45,42	73,25	222,1	277,2
	Energia [J]	0,007	0,011	0,059	0,076
	Comunicação	34%	18%	20%	28%
	Execução	66%	82%	80%	72%

No sistema *desktop*, os resultados revelam uma clara hierarquia de desempenho. A implementação VTM_{SIMD} é a mais rápida, superando tanto a versão

escalar quanto a GPU. Embora a GPU possua um poder de processamento massivo, seu desempenho é limitado pela sobrecarga de comunicação (transferência de dados entre CPU e GPU), que consumiu em média 84% do tempo total da etapa. Ainda assim, a GPU foi mais rápida que a implementação escalar base $VTM_{escalar}$.

No sistema embarcado, o cenário é diferente. Embora seja mais lento em termos de velocidade bruta, o Jetson TX2 demonstra uma eficiência energética notável. A CPU ARM do Jetson consumiu aproximadamente 38% menos energia que a implementação mais eficiente do desktop (VTM_{SIMD}). Sua GPU integrada foi ainda mais eficiente, consumindo 48% menos energia. Outro ponto importante é que, na GPU integrada, o gargalo não é a comunicação (que é rápida devido à memória compartilhada com a CPU), mas sim o tempo de execução do kernel.

Essas descobertas destacam um *trade-off* fundamental: a plataforma desktop oferece desempenho bruto superior, enquanto o sistema embarcado se destaca pela eficiência energética, tornando-o ideal para aplicações com restrições de energia.

4. CONCLUSÕES

Este trabalho apresentou uma avaliação de desempenho e consumo de energia da etapa de classificação do ALF no VVC, comparando implementações em CPU (escalar e SIMD) e GPU em plataformas desktop e embarcada.

Os resultados demonstraram que, em um *desktop* de alto desempenho, a implementação VTM_{SIMD} oferece a maior velocidade e boa eficiência energética, pois a aceleração em GPU dedicada é severamente prejudicada pela sobrecarga de comunicação. Em contraste, a plataforma embarcada Jetson TX2, embora mais lenta, oferece uma eficiência energética muito superior, com sua GPU integrada sendo limitada pela capacidade de processamento, e não pela transferência de dados. As conclusões ressaltam que a estratégia de implementação ótima para o ALF não é universal, mas altamente dependente da plataforma e dos requisitos da aplicação, exigindo uma análise cuidadosa do *trade-off* entre velocidade de processamento e consumo de energia.

5. REFERÊNCIAS BIBLIOGRÁFICAS

Bossen, F. et al. (2020). “JVET-T2010: VTM common test conditions and software reference configurations for SDR video.” Doc. JVET.

Bossen, F. et al. (2021). “Vvc complexity and software implementation analysis,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3765–3778.

Bross, B. et al. (2021). “Overview of the versatile video coding (vvc) standard and its applications,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3736–3764.

Fabricio, V., Storch, I., & Palomino, D. (2025). “Processing Time Evaluation of the Classification Step in the Adaptive Loop Filter of VVC under Multiple Programming Paradigms.” 2025 IEEE 16th Latin America Symposium on Circuits and Systems (LASCAS).

Fang, J. & Yang, F. (2023). "The Optimization of Adaptive Loop Filter Based on the Reduction of the Filter Sets," 2023 International Conference on Ubiquitous Communication (Ucom).

Farhat, I. et al. (2022). “Efficient HW Design of Adaptive Loop Filter for 4k ASIC VVC Encoder,” 2022 Picture Coding Symposium (PCS).

JVET. (2023). VVC Test Model Encoder (VTM). Disponível em: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM.

Karczewicz, M. et al. (2021). “Vvc in-loop filters,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 10, pp. 3907–3925.

Saha, A. et al. (2023). "Performance analysis of optimized versatile video coding software decoders on embedded platforms," Journal of Real-Time Image Processing.

Yin, W., Zhang, K., & Zhang, L. (2023). "Extended Adaptive Loop Filter Beyond VVC," 2023 IEEE International Conference on Visual Communications and Image Processing (VCIP).