Avaliação do Tempo de Processamento da Etapa de Classificação no Filtro de Loop Adaptativo do VVC em Múltiplos Paradigmas de Programação

VITÓRIA DE SOUZA FABRICIO[1]; IAGO STORCH[2]; DANIEL PALOMINO[3]

[1]*Universidade Federal de Pelotas – vitoria.sfabricio@gmail.com,*
[2]*Universidade Federal de Pelotas – icstorch@inf.ufpel.edu.br*
[3]*Univerisidade Federal de Pelotas – dpalomino@inf.ufpel.edu.br*

1. INTRODUÇÃO

As aplicações de codificação de vídeo exigem grande poder de processamento para alcançar compressão eficiente (Bossen, 2021). Para atender à crescente demanda por eficiência, novos padrões foram introduzidos, como o Versatile Video Coding (VVC), que oferece melhorias significativas na compressão através de ferramentas inovadoras (Bross, 2021). Uma dessas tecnologias é o Filtro de Loop Adaptativo (ALF), que melhora a eficiência da codificação, mas aumenta o esforço computacional (Karczewicz, 2021).

O ALF aplica filtragem espacial nas amostras reconstruídas para mitigar artefatos de codificação. Cada bloco de dimensões 4x4 é classificado de acordo com a direcionalidade e atividade laplaciana das suas texturas, e um conjunto de parâmetros de filtro é derivado para maximizar a qualidade percebida (Karczewicz, 2021). O ALF foi originalmente proposto para o padrão HEVC, mas devido a sua alta complexidade, ele não foi incorporado ao padrão.

No entanto, apesar da sua alta complexidade, o ALF foi definitivamente incorporado ao padrão VVC. Embora existam métodos para reduzir o tempo de processamento do ALF (Han, 2020; Luo, 2015), estes métodos se concentram em mudanças algorítmicas dentro de um paradigma de programação puramente sequencial tradicional. Entretanto, tendo em vista que o ALF é aplicado sobre as amostras já reconstruídas e não há dependências de dados, ele apresenta uma enorme possibilidade para exploração de paralelismo.

Sendo assim, este trabalho avalia o tempo de processamento do ALF em três paradigmas de programação: um processamento sequencial tradicional usando CPU, um que utiliza instruções SIMD e outro que emprega computação em GPU com a tecnologia CUDA. A comparação entre esses paradigmas visa demonstrar a eficácia de cada um no processamento da classificação do ALF, orientando futuros desenvolvimentos de técnicas de aceleração.

2. METODOLOGIA

O ALF do VVC melhora a qualidade da imagem ao ajustar a filtragem de blocos de pixels. 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 4x4 amostras. No entanto, para determinar a classe de cada bloco, uma região de 8x8 amostras centrada no bloco é considerada. Cada bloco é classificado em uma de 25 classes

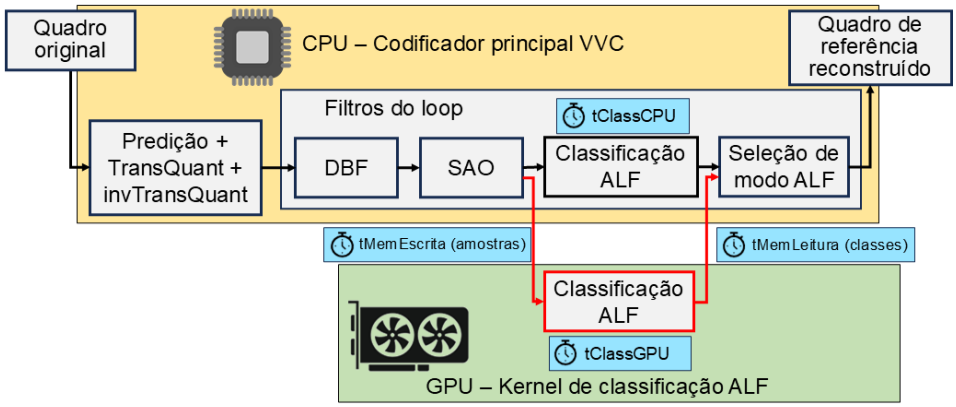

Figura 1. Visão geral do sistema de codificação CPU+GPU. Os contornos em vermelho destacam o desvio da GPU utilizado para a classificação ALF.

de acordo com a equação (1). Este cálculo é baseado da direção de textura D e na atividade de textura local quantizada Ã.

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, considerando um halo de duas amostras 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.

$$\boldsymbol{C = 5D + \tilde{A}} \quad (1)$$

A classificação ALF processa blocos de 4x4 de forma independente, permitindo grande potencial de paralelização. Três paradigmas de programação foram avaliados: processamento sequencial em CPU, paralelismo SIMD em CPU, e paralelismo massivo em GPU. Todos os três métodos geram os mesmos resultados, sem variação de eficiência de codificação.

O padrão VVC conta com um codificador de software 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

Tabela 1. Tempo médio de classificação ALF por quadro [ms].

Vídeo	VTM Escalar	VTM SIMD	GPU	
			Com.	Proc.
BasketballDrive	6,159	1,514	3,460	0,763
BQTerrace	6,195	1,493	3,321	0,670
Cactus	6,374	1,507	3,334	0,773
MarketPlace	6,398	1,515	3,352	0,766
RitualDance	6,391	1,500	3,392	0,766
Média	**6,374**	**1,507**	**3,352**	**0,766**

durante o cálculo do gradiente. Ainda assim, os blocos 4×4 são processados de forma sequencial.

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.

3. RESULTADOS E DISCUSSÃO

Os resultados experimentais para as implementações baseadas em CPU — VTM escalar e VTM SIMD — foram obtidos utilizando o VTM 23.4. Os kernels de classificação ALF desenvolvidos foram usados para obter o tempo de processamento na GPU. As partes da Figura 1 destacadas em azul indicam os pontos do processo onde o tempo é medido. No caso das implementações em CPU, apenas o tempo de execução da classificação ALF é considerado, representado por tClassCPU na Figura 1. Já no processamento com GPU, há a necessidade de tarefas extras de comunicação entre CPU e GPU para trocar as amostras de quadros e as classes resultantes. O tempo de processamento na GPU inclui o tempo de execução do kernel (tClassGPU na Figura 1), além do tempo para mover as amostras dos quadros para a memória da GPU (tMemEscrita na Figura 1) e para transferir as classes de volta para a memória da CPU (tMemLeitura na Figura 1).

Os experimentos de CPU foram realizados em um computador com processador Intel Core i7-8700, 16 GB de RAM e sistema Ubuntu 20.04, com o VTM utilizando o conjunto de instruções AVX2 SIMD. Os experimentos de GPU foram realizados em uma placa gráfica NVIDIA RTX 3080.

Independente do paradigma de programação, os primeiros 32 quadros de sequências de vídeo Full HD (1920x1080) foram processados, utilizando os parâmetros de quantização recomendados pelas Condições de Teste Comum do VVC (CTC): 22, 27, 32 e 37. O tempo de classificação ALF foi calculado em média para todos os quadros e QPs. A Tabela 1 apresenta os tempos médios por quadro, em milissegundos, para os três paradigmas de programação considerados, com a

coluna "Com" representando o tempo de comunicação entre CPU e GPU e "Proc" representando o tempo de processamento real na GPU.

A primeira conclusão da Tabela 1 é que, para um paradigma específico, todos os vídeos requerem um tempo de processamento similar, devido à regularidade do procedimento de classificação ALF, que independe do conteúdo do vídeo. Comparando os diferentes paradigmas, o VTM SIMD é muito mais rápido que o VTM escalar, com 1,507 ms contra 6,374 ms por quadro, em média. A GPU, considerando o tempo de comunicação e processamento, requer cerca de 4,118 ms para processar um quadro, sendo mais rápida que o VTM escalar, mas mais lenta que o VTM SIMD.

O desempenho intermediário da GPU se deve ao grande *overhead* de comunicação. A classificação na GPU leva apenas 0,766 ms, cerca de duas vezes mais rápido que o VTM SIMD, mas a movimentação de dados entre a CPU e a GPU adiciona cerca de 3,352 ms por quadro. Caso mais tarefas fossem executadas na GPU, o paradigma de GPU poderia se tornar uma solução mais promissora. Esses resultados mostram que as oportunidades de paralelização da classificação ALF podem ser exploradas de forma diferente dependendo do paradigma de computação escolhido, destacando a importância de usar as plataformas de computação de forma eficiente, conforme sua disponibilidade e a carga de trabalho.

4. CONCLUSÕES

Este trabalho avaliou o tempo de processamento da classificação no ALF do VVC utilizando três paradigmas de programação: implementações escalares e SIMD em CPU, e uma implementação massivamente paralela em CUDA para GPU. Os resultados mostraram que o VTM SIMD supera o VTM escalar, evidenciando as vantagens do paralelismo SIMD. No entanto, embora a GPU tenha mostrado ganhos em velocidade de processamento, seu desempenho foi inferior ao VTM SIMD devido ao alto custo de comunicação entre CPU e GPU. O estudo destaca a importância de escolher o paradigma de programação mais adequado à plataforma de computação.

5. REFERÊNCIAS BIBLIOGRÁFICAS

BOSSEN, F. et al. Vvc complexity and software implementation analysis. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 31, n. 10, p.3765–3778, 2021.

BROSS, B. et al. Overview of the versatile video coding (vvc) standard and its applications. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 31, n. 10, p. 3736 3764, 2021.

HAN, X. et al. Optimization of motion compensation based on gpu and cpu for vvc decoding. **2020 IEEE International Conference on Image Processing (ICIP)**. Abu Dhabi: IEEE, 2020. p. 1196–1200.

JVET. VTM VVC **Reference Software**. 2023. Acessado em 08 de Outubro de 2024. Disponível em: https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware\ VTM.

KARCZEWICZ, M. et al. Vvc in-loop filters. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 31, n. 10, p. 3907–3925, 2021.

LUO, F. et al. Multiple layer parallel motion estimation on gpu for high efficiency video coding (hevc). **2015 IEEE International Symposium on Circuits and Systems (ISCAS)**. Lisbon: IEEE, 2015. p. 1122–1125.