ALGORITMO DE PARTICIONAMENTO DINÂMICO DE TILES BASEADO EM HISTÓRICO PARA GANHO DE PARALELISMO NO PADRÃO DE COMPRESSÃO DE VÍDEO HEVC

IAGO STORCH; BRUNO ZATT, DANIEL PALOMINO, LUCIANO AGOSTINI
Universidade Federal de Pelotas – Grupo de Arquiteturas e Circuitos Integrados
{icstorch, zatt, dpalomino, agostini}@inf.ufpel.edu.br

1. INTRODUÇÃO

Com os avanços tecnológicos e a popularização de *smartphones* e serviços de *streaming*, vídeos digitais estão cada vez mais presentes no cotidiano da população. Entretanto, a transmissão e armazenamento dos vídeos digitais se tornam inviáveis quando esses vídeos estão em seu formato original, e nesse contexto surgem os compressores de vídeos digitais.

A compressão de vídeo é realizada através da exploração de redundâncias temporais e espaciais, e através desse processo é possível atingir altas taxas de compressão, mas criando uma grande carga de trabalho para o processador (SULLIVAN, 2012). O padrão HEVC herdou uma série de ferramentas do seu antecessor, além de inserir muitas outras, e através disso atingiu uma taxa de compressão de 36% a 70% maior do que o H.264/AVC (OHM, 2012). Esse aumento de eficiência, no entanto, veio ao custo de um aumento significativo de complexidade (OHM, 2012).

Buscando amortizar esse aumento de complexidade, o padrão HEVC foi desenvolvido buscando suporte ao processamento paralelo, e esse suporte se dá em parte pela inserção de uma nova ferramenta chamada de *Tiles*. Os *Tiles* são uma ferramenta que divide a imagem em porções retangulares, compostas por um número inteiro de *Coding Tree Units* (CTUs), de modo que cada *Tile* possa ser processado de forma completamente independente, logo sendo altamente suscetível à paralelização.

Essas porções retangulares são delimitadas por *boundaries*, que são linhas verticais e horizontais que cortam a imagem e podem dividir um quadro de forma uniforme ou não-uniforme. Essa configuração da quantidade e tamanho dos *Tiles* deve ser feita num momento anterior ao início da compressão, e uma vez configurado, o mesmo particionamento será utilizado para todos os quadros.

Como o processo de compressão de vídeo explora redundâncias, áreas homogêneas e estáticas são comprimidas significativamente mais rápido do que áreas heterogêneas e com alta movimentação, logo, uma mesma quantidade de dados pode demorar mais ou menos tempo para ser comprimida de acordo com seu conteúdo. Com base nessa constatação, pode-se dizer que o método de funcionamento estático dos *Tiles* não reflete o comportamento dinâmico dos vídeos, e não explora o paralelismo da melhor maneira possível (STORCH, 2015).

Sendo assim, o objetivo deste trabalho é desenvolver um algoritmo capaz de realizar um particionamento de *Tiles* dinâmico, selecionando o melhor particionamento para cada quadro buscando obter uma distribuição homogênea de carga de trabalho entre os *Tiles*, e deste modo, aumentando o paralelismo proveniente destes.

2. METODOLOGIA

Para realizar o desenvolvimento do algoritmo proposto, o *software* de referência do padrão HEVC, o *HEVC Test Model* (HM-16.0), foi utilizado como

base. O primeiro passo foi identificar em que ponto do código a ferramenta de *Tiles* é implementada, e alterar seu funcionamento para garantir que fosse possível aplicar particionamentos diferentes para cada quadro. Em seguida, foi inserida uma sub-rotina no código para capturar o tempo de compressão de cada CTU, visto que essa é a menor porção manipulável dentro de um *Tile*, e o somatório do tempo de compressão das CTUs dentro de um *Tile* representa o tempo de compressão total do respectivo *Tile*.

Em seguida, uma série de vídeos foram comprimidos e os tempos de compressão por CTU foram extraídos para servir como referência da carga computacional das mesmas, e ao analisar esses tempos, foi possível perceber que os quadros sucessivos em um vídeo tem uma distribuição de carga de trabalho muito semelhante entre si. Sendo assim, constatou-se que a distribuição da carga de trabalho não se altera significativamente ao longo dos quadros de um vídeo, logo, seria possível analisar os quadros anteriores para prever como os próximos iriam se comportar, e com base nisso definir um particionamento de *Tiles* que distribuísse de maneira mais homogênea a carga de trabalho entre os *Tiles*.

Com base nessa descoberta, foi desenvolvido um algoritmo de particionamento dinâmico baseado em histórico que, com base numa matriz que representa a distribuição da carga de trabalho dos quadros anteriores, prevê a distribuição da carga de trabalho do próximo quadro e gera o particionamento que resulta no melhor *speedup*. Como no primeiro quadro não se tem referência para quadros anteriores, o primeiro quadro é comprimido com o particionamento uniforme padrão e o tempo de compressão por CTU é salvo na matriz de distribuição de carga de trabalho. Para definir os *boundaries* dos próximos quadros, o histórico de distribuição da carga de trabalho dos quadros anteriores é usado como referência para prever a distribuição de carga de trabalho para o quadro atual.

O Algoritmo 1 descreve os passos da solução proposta. Para calcular o nível de balanceamento da distribuição de carga computacional entre os *Tiles* foi utilizada a métrica s*peedup.* O *speedup* pode variar de 1 ao número de *Tiles*, com um *speedup* igual 1 representando o *speedup* mínimo, o caso em que todo o trabalho está concentrado num único *Tile*, e um *speedup* igual ao número de *Tiles* representando o *speedup* máximo, o caso em que o trabalho está dividido igualmente entre os *Tiles*. A equação (1) descreve a equação para o cálculo do *speedup*, onde $\mathbf{Ttile_i}$ representa o tempo para comprimir o **i-ésimo** *Tile*, e **n** representa o número total de *Tiles*. Já para estimar o tempo de compressão de um possível *Tile* foi utilizada a equação (2), que representa o somatório dos tempos de compressão de todas as CTUs dentro de um *Tile* no histórico de distribuição de

Algoritmo 1 – Algoritmo de particionamento dinâmico

Entrada: matriz de histórico de distribuição de carga **Mij**, número de *Tiles* verticais **V**, número de *Tiles* horizontais **H**

```
1:  Vb ← V -1
2:  média ← cargaTotal/V
3:  para cada v ∈ Vb faça:
4:    para cada linha ∈ Mij faça:
5:      atual ← soma do workload das linhas
6:      se atual ≈ média×v
7:        selecionar linha como v-ésimo boundary
8:  Hb ← H – 1
9:  média ← cargaTotal/H
10: para cada h ∈ Hb faça:
11:   para cada coluna ∈ Mij faça:
12:     atual ← soma do workload das colunas
13:     se atual ≈ média×h
14:       selecionar coluna como h-ésimo boundary
```

$$Speedup = \frac{\sum_{i=0}^{n-1} Ttile_i}{max(Ttile_0, \ldots, Ttile_{n-1})} \quad (1)$$

$$Atual = \sum_{i=0}^{linha|col\ (largura|altura)EmCTU} \sum_{j=0} Histórico_{ij} \quad (2)$$

carga, sendo que na linha 5 do algoritmo a equação usa **linha** e **larguraEmCTU** como limites superiores e na linha 12 usa **col** e **alturaEmCTU**.

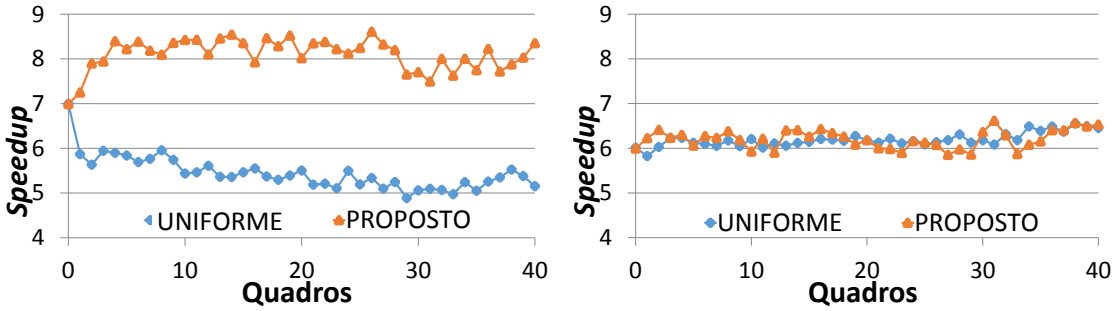
3. RESULTADOS E DISCUSSÃO

Após desenvolver o algoritmo, foi necessário validar seus resultados, e para isso foram consideradas duas métricas: (1) *speedup,* para avaliar o quanto de paralelismo o algoritmo foi capaz de proporcionar, e (2) BD-Rate (BJØNTEGAARD, 2001), que indica a perda de eficiência de compressão de uma técnica em relação a outra. Considerando essas métricas, uma série de vídeos foram selecionados e submetidos a três conjuntos de testes: no primeiro conjunto, os vídeos foram comprimidos usando o compressor original, sem a utilização de *Tiles*, para obter dados referentes à eficiência de compressão e tempo de processamento; no segundo conjunto os vídeos foram comprimidos usando o compressor original, mas utilizando *Tiles* uniformes para descobrir qual é o *speedup* e a perda de BD-Rate convencional relacionados ao uso de *Tiles* uniformes; no terceiro conjunto de testes, os vídeos foram comprimidos utilizando o compressor com o algoritmo inserido e particionamentos não-uniformes para descobrir o quanto o algoritmo foi capaz de melhorar o *speedup*, e se houve alguma mudança significativa de BD-Rate.

O compilado de todos os resultados de *speedup* e BD-Rate está agrupado por resolução e número de *Tiles* na Tabela 1. Analisando esta tabela, é possível observar que o algoritmo implementado foi capaz de aumentar o *speedup* proporcionado pela ferramenta de *Tiles*, em média, 3%, 10,01% e 7,52% para os particionamentos 2x2, 3x3 e 4x4, respectivamente. Dentre esses resultados, é importante ressaltar que o maior ganho obtido foi 38% de *speedup* para o vídeo *FourPeople* com o particionamento 3x3, como mostrado na Fig. 1 (a). Esse resultado se deve ao fato do vídeo *FourPeople* ter uma distribuição de carga computacional bastante heterogênea e sua concentração ser bastante propícia a um particionamento 3x3. Já o pior resultado foi uma queda de *speedup* de 1,07% no vídeo *RaceHorses* com um particionamento 3x3, como mostrado em Fig. 1 (b). Esse resultado se deve, principalmente, à baixa taxa de quadros por segundo do

Tabela 1 – Resultados de *Speedup* e BD-Rate

Tiles	Uniforme		Proposto		Ganho	
	Speedup	BD-Rate	*Speedup*	BD-Rate	*Speedup*	BD-Rate
1920x1080						
2x2	3.4901	0.68%	3.723	0.64%	6.67%	-0.04
3x3	6.9987	1.43%	7.6365	1.51%	9.11%	0.09
4x4	11.5111	2.27%	12.6826	2.27%	10.18%	0.00
1280x720						
2x2	3.6416	2.79%	3.68538	2.70%	1.20%	-0.09
3x3	6.79533	5.33%	7.70983	5.40%	13.46%	0.08
4x4	11.8728	8.87%	12.4498	8.74%	4.86%	-0.13
832x480						
2x2	3.4423	0.87%	3.48058	0.88%	1.11%	0.02
3x3	6.14668	1.91%	6.60485	2.00%	7.45%	0.09
Média						
2x2	3.5247	1.44%	3.6297	1.41%	3.00%	-0.04
3x3	6.6469	2.89%	7.3171	2.97%	10.01%	0.09
4x4	11.6919	5.57%	12.5662	5.50%	7.52%	-0.07



(a) *FourPeople* – *Tiles* 3x3 (b) *RaceHorses* – *Tiles* 3x3

Fig. 1 – Resultados de *speedup* no tempo

vídeo, uma característica que reduz a redundância temporal do quadros e faz com que quadros sucessivos tenham distribuições de carga computacional distintas, logo, reduzindo a precisão da carga computacional prevista pelo algoritmo.

Em relação ao BD-Rate, pode-se notar que alterar do particionamento uniforme para um que visa o maior *speedup* não causa grandes impactos na eficiência de compressão. Em média, houve uma redução de 0,02% de BD-Rate, mas como todos os resultados foram próximos de zero e houveram tanto ganhos quanto perdas de BD-Rate, pode-se dizer que o impacto na eficiência de compressão é desprezível. Além destes dados, o tempo de execução do algoritmo inserido foi medido, e constatou-se que ele representa menos do que 10^{-4}% do tempo total de execução do compressor, o que pode ser considerado um *overhead* de tempo desprezível.

4. CONCLUSÕES

Este trabalho apresentou um algoritmo de particionamento de *Tiles* dinâmico baseado em histórico para prever a distribuição de carga computacional de quadros futuros e com base nisso, gerar um particionamento de *Tiles* específico para cada quadro visando o maior ganho de *speedup* possível. Com base nos resultados apresentados, é possível afirmar que o algoritmo obteve um ganho de *speedup* considerável quando comparado ao particionamento uniforme, com uma variação de BD-Rate e *overhead* de tempo desprezíveis.

5. REFERÊNCIAS BIBLIOGRÁFICAS

SULLIVAN et al. Overvew of the High Eficiency Video Coding (HEVC) Standard; **IEEE Transactions on Circuits and Systems for Video Technology**. [S.l:s.n], v. 22 n.12, p.1649-1668, 2012.

OHM et al. Comparison of the Coding Efficiency of Video Coding Standsrds – Including High Efficiency Video Coding (HEVC). **IEEE Transactions on Circuits and Systems for Video Technology**. [S.l:s.n], v. 22 n.12, p. 1669-1684, 2012.

STORCH et al. Avaliação do Particionamento de Quadros Usando Tiles para a Compressão de Vídeo Paralela Segundo o Padrão HEVC. **XXIV Congresso de Iniciação Científica**. Pelotas, 2015. **Anais ...**

BJØNTEGAARD, G. **Calculation of Average PSNR Differences between RD-curves**. ITU Documents, Austin 2011. Acessado em 05 julho de 2016, Online. Disponível em: wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-M33.doc