AVALIAÇÃO DA HIERARQUIA DE MEMÓRIA EM GPU PARA ALGORITMOS DE ESTIMAÇÃO DE MOVIMENTO UTILIZANDO OPENCL

MATEUS MELO; HENRIQUE MAICH; LUCIANO AGOSTINI; BRUNO ZATT; MARCELO PORTO

Universidade Federal de Pelotas – Grupo de Arquitetura e Circuitos Integrados
Centro de Desenvolvimento Tecnológico
{msdmelo, hdamaich, agostini, zatt, porto}@inf.ufpel.edu.br

1. INTRODUÇÃO

Atualmente, a grande maioria dos dispositivos móveis presentes no mercado possui suporte à captura e manipulação de vídeos digitais. Estes dispositivos possuem limitações em bateria e armazenamento, assim se torna inviável o armazenamento ou transmissão dos vídeos digitais sem compressão durante sua captura, devido à alta quantidade de dados. Para viabilizar a manipulação de vídeo é utilizado um codificador, com objetivo de representar a mesma informação de forma comprimida.

O padrão de codificação de vídeo estado da arte é o HEVC (High Efficient Video Coding) (ITU-T, 2013), sendo o padrão que atinge as maiores taxas de compressão para uma mesma qualidade de imagem, em comparação aos demais padrões de compressão existentes. Porém, para obter estas altas taxas de compressão, as etapas presentes no processo de codificação tornaram-se mais complexas. Dentre elas destaca-se a Estimação de Movimento (ME – Motion Estimation), a qual é responsável pelos maiores ganhos em compressão, porém, é a responsável pelo maior percentual de tempo de codificação.

A ME é responsável por identificar e remover as redundâncias presentes em quadros temporalmente vizinhos, assim gerando uma codificação mais eficiente. A ME trabalha em nível de blocos dentro do quadro a ser codificado, assim para um determinado bloco, é realizada a busca pelo bloco mais similar (melhor casamento) dentro de uma área de busca em um quadro já codificado (quadro de referência) (PORTO, 2008).

Existem diversos algoritmos para a realização desta etapa, de forma a obter melhores resultados de acordo com o padrão de busca empregado. Os algoritmos de ME são classificados como: ótimo e rápidos. O algoritmo Full Search (FS) é o algoritmo ótimo, pois avalia todos os blocos candidatos dentro da área de busca, assim, encontrando sempre o melhor resultado.

Os algoritmos chamados rápidos possuem uma heurística que acelera a escolha do melhor bloco candidato, devido considerar um menor número de blocos candidatos, podendo assim não encontrar o melhor resultado dentro da área de busca. Um exemplo de algoritmo rápido é o Diamond Search (DS) (PORTO, 2008), onde a busca pelo melhor bloco candidato é feita de forma iterativa.

Dentre os algoritmos rápidos, existem variações do algoritmo FS, utilizando sub-amostragem de pixels ou de blocos candidatos. A sub-amostragem de bloco reduz o número de blocos candidatos a serem comparados dentro da área de busca Nesse trabalho, será utilizada a sub-amostragem 2:1, ou seja, aquela onde metade dos blocos candidatos presentes na área de busca são comparados. Os algoritmos FS utilizando este tipo de sub-amostagem receberão o sufixo “21” neste trabalho (FS21).

Neste trabalho, o cálculo do SAD foi utilizado como critério de similaridade para identificar o melhor bloco candidato devido a sua paralelização, uma vez que há independência de dados entre os blocos, além de ser o critério mais utilizado na literatura, devido sua simples implementação em hardware. Grande parte dos

processadores embarcados nos dispositivos móveis atuais possuem GPUs com suporte à programação paralela em OpenCL (Khronos, 2011). Desta forma, etapas como a ME de um codificador de vídeo podem ser paralelizadas utilizando essas GPUs, presentes nos processadores embarcados.

O OpenCL mapeia essa hierarquia nos seguintes níveis: global, constant, local e private. As memórias global e contant são mapeadas para DDR3 da GPU, a memória local é mapeada para Cache L1/Scratchpad, e a memória private é mapeada pra registradores, classificação feita de acordo com sua capacidade de armazenamento e sua velocidade de processamento/acesso.

Sendo assim, este trabalho explora a análise de algoritmos paralelos de ME utilizando o OpenCL, avaliando o desempenho dos algoritmos em diferentes níveis da hierarquia de memória de GPUs, além da comunicação entre as memórias (utilizada pela CPU e da GPU). Essas avaliações foram realizadas, pois conforme apresentado em Wolf (2013), a comunicação com a memória está entre os principais fatores de desempenho e eficiência energética em sistemas embarcados.

2. METODOLOGIA

Para a execução desses algoritmos na GPU é necessário que os dados a serem utilizados sejam carregados na memória da GPU. Isto é feito através de instruções de transferência entre a memória da CPU e da GPU, sendo que esta transferência de dados entre as memórias impacta diretamente no consumo energético desses dispositivos. Desta forma, foi verificado o impacto energético sobre a quantidade de dados a serem transferidos para a GPU, utilizando duas abordagens. Na primeira, todos os blocos a serem codificados são transferidos junto com a sua área de busca para a memória da GPU. Entretanto, existem regiões da área de busca que são compartilhadas entre os blocos vizinhos, aumentando assim a quantidade de dados a serem escritos na memória. Os algoritmos com base nessa abordagem receberam o sufixo “SA” (Search Area) neste trabalho (FSSA, FS21SA, DSSA).

Para evitar essa reescrita, foi realizada uma segunda abordagem, na qual todo o quadro de referência é enviado, juntamente com o primeiro bloco a ser codificado do quadro, e na execução da GPU é definida a área de busca correspondente para os próximos blocos. Assim, não é necessário o reenvio de amostras compartilhadas por outros blocos, pois sua área de busca já está presente na memória da GPU. Os algoritmos com base nessa abordagem receberam o sufixo “FF” (Full Frame), neste trabalho (FSFF, FS21FF, DSFF).

O primeiro experimento verificou a quantidade de dados transferidos entre as memórias, para cada um dos algoritmos implementados e suas respectivas abordagens. Já para o segundo experimento, foi avaliado o tempo médio de execução para cada implementação.

Os resultados dos dois primeiros experimentos serviram como base para o terceiro experimento, que consiste na avaliação da redução no tempo médio de execução dos algoritmos utilizando diferentes níveis da hierarquia de memória da GPU. Para isso, os algoritmos foram modificados para utilizar a memória local para o armazenamento dos SADs de todos os blocos candidatos, que antes estavam armazenados na memória global. Dentre os algoritmos, somente a versão FF foi utilizada para o terceiro experimento, pois possuírem uma menor transferência de dados entre as memórias e um tempo de execução similar ou inferior aos algoritmos SA.

Para a realização desses experimentos foram utilizados blocos de tamanho 8x8, devido ao fato que com esse tamanho de bloco não há uma grande perda na

qualidade visual do vídeo após a codificação. Nas diferentes versões do algoritmo FS foi utilizado uma área de busca de tamanho 64x64 amostras e, para o algoritmo DS, foi definido a realização de quatro iterações (remetendo a uma área de busca 26x26 amostras), pois, conforme PORTO (2008), este é o numero médio de iterações realizadas em vídeos SD (720x480 amostras).

As simulações foram realizadas em um computador com processador Intel Core i7-3770 @ 3.4 GHz e uma GPU NVidia GeForce GT 630 com suporte à programação OpenCL 1.1, utilizando 12 sequências de teste, sendo, 4 vídeos WVGA (416x240 amostras), 5 vídeos HD 720p (1280x720 amostras) e 5 vídeos HD 1080p (1920x1080 amostras).

3. RESULTADOS E DISCUSSÃO

A Tabela 1 apresenta os resultados médios de volume de transferência de dados, em MB, entre as memórias da CPU e GPU para os algoritmos avaliados, bem como os tempos médios de execução de cada algoritmo em cada uma das classes de vídeo analisadas.

O primeiro experimento foi realizado com o intuito de avaliar a transferência de dados entre as memórias (da CPU e da GPU). Conforme pode ser observado nas colunas “Dados” da Tabela 1, os algoritmos com o sufixo FF têm uma menor transferência de dados, para todas as resoluções de vídeos utilizadas nesse trabalho, em relação aos algoritmos com sufixo SA, isso devido às áreas de busca entre blocos vizinhos não serem reescritas, como citado anteriormente.

Tabela 1: Resultados dos experimentos

Algoritmo	WVGA		HD 720p		HD 1080p	
	Dados(MB)	Tempo(s)	Dados(MB)	Tempo(s)	Dados(MB)	Tempo(s)
FSSA	417,04	14,69	3849,61	133,51	8661,62	304,75
FSFF	57,13	13,54	527,34	123,61	1186,52	283,01
FS21SA	394,19	10,33	3638,67	95,18	8187,01	214,95
FS21FF	34,28	9,21	316,41	85,24	711,91	192,53
DSSA	66,77	6,89	616,33	64,63	1386,75	148,41
DSFF	12,14	6,93	112,06	64,17	252,14	147,85

O segundo experimento avaliou o tempo médio de execução dos algoritmos. Como pode ser observado, a melhor opção para os algoritmos FS foi a abordagem com sufixo FF, já que eles possuem um menor tempo de execução. Porém, quando avaliamos os algoritmos DS, a melhor opção para vídeos de baixa resolução é a abordagem com sufixo SA, e para as demais coincide com os resultados obtidos para os algoritmos FS. Esta inconformidade pelo fato que no algoritmo DSFF o tempo inicial para a GPU definir a área de busca acaba se tornando mais visível no resultado de tempo de execução, para os demais algoritmos com abordagem FF esse tempo acaba por ser amortizado pelos cálculos realizados pela GPU.

Os resultados do terceiro experimento, onde a memória local da GPU foi utilizada para o armazenamento dos SADs dos algoritmos com sufixos FF, pois possuem uma menor quantidade de dados transferidos, estão apresentados na Tabela 2. Assim, observou-se que, os algoritmos se mostraram mais eficientes quando mais blocos candidatos são processados, assim, os maiores ganhos em se usar a memória local foram obtidos nos algoritmos FF, além disso, pequenas perdas de desempenho podem ser observadas para o algoritmo DS, devido à organização dos dados da memória local, onde a organização se dá em banco de registradores. Sendo nessa organização o acesso a esses bancos realizado sequencialmente, assim, o baixo número de SADs de blocos candidatos acaba

causando várias colisões no acesso dos dados. Esse número de colisões ao acessa ao banco de registradores acaba reduzindo o desempenho geral do algoritmo em relação à memória global no qual o acesso de dados é realizado em bloco.

Tabela 2: Redução de tempo de execução dos algoritmos utilizando memória *local*

Algoritmo	WVGA	HD 720p	HD 1080p
FSFF(%)	18,40	17,59	18,80
FS21FF(%)	3,81	4,13	3,52
DSFF(%)	-3,37	-2,76	-3,07

De modo geral, o tempo de execução dos algoritmos é proporcional ao aumento da resolução dos vídeos, devido a maiores quantidades de dados a serem processados, proporcionalidade que pode ser observada em todos os experimentos realizados. E também, foram obtidos maiores reduções no tempo de processamento utilizando memória local com maiores quantidades de dados a serem processados quando comparados utilizando a memória global.

4. CONCLUSÕES

Este trabalho apresentou uma avaliação de algoritmos de ME utilizando GPUs com suporte à programação OpenCL. Foram implementados três algoritmos de ME, utilizando duas abordagens de transferência de dados para a memória da GPU, uma enviando a área de busca e outra enviando o quadro completo. Assim, baseado nos algoritmos desenvolvidos, foram avaliadas diversas abordagens e execução de algoritmos de ME em GPUs, como a transferência de dados para a memória da GPU e tempo de execução destes algoritmos para diferentes níveis de hierarquia de memória.

Conforme os experimentos, a abordagem FF se torna eficiente para dispositivos embarcados, pois reduz a quantidade de dados transferidos entre as memórias da CPU e da GPU, sendo esta transferência um dos pontos com grande impacto energético neste tipo de dispositivos. Além de algoritmos FS se mostraram propícios a uso de memória local, assim reduzindo o tempo de execução destes algoritmos.

A partir desses resultados, tem-se como objetivo a implementação de algoritmos de ME, eficiente para dispositivos embarcados, utilizando vários tamanhos de blocos. Além disso, pretendemos expandir os resultados utilizando processadores embarcados nos quais conseguimos medir consumo energético e desta forma avaliar o algoritmo desenvolvido.

5. REFERÊNCIAS BIBLIOGRÁFICAS

ITU-T, International Telecommunication Union. **Recommendation ITU-T H.265: High efficiency video coding**. Abril, 2013.

WOLF. W, K. M. “Memory System Optimization of Embedded Software”. **Proceedings of the IEEE**, 2013

KHRONOS, O.W.G.. The OpenCL Specification. Revision 44.ed. [S.1.]: Khronos Group, 2011.

PORTO M., et. al., “Architectural Desing for the New QSDS with Dynamic Iteration Control Motion Estimation Algorithm Targeting HDTV”, **21st SBCCI**, p. 216-221, 2008.