Aumento de Desempenho de Codificação Intra H.264 Usando Arquitetura de GPU NVIDIA CUDA®

Ronaldo Husemann
DELET - UFRGS
Osvaldo Aranha, 103
Porto Alegre,RS
+55 51 33083515
rhusemann@inf.ufrgs.br

Anderson Giacomolli, Augusto Lenz, Diego Schwingel
CETEC - UNIVATES
Av. Avelino Tallini, 171
{a.giacomolli, augustollenz, diego.schwingel}@gmail.com

Valter Roesler
Inst. Informática - UFRGS
Bento Gonçalves, 9500
Porto Alegre, RS
+55 51 33086167
roesler@inf.ufrgs.br

ABSTRACT

The recent growth of high definition multimedia applications imposes several restrictions to traditional ISO MPEG-x e ITU-T H.26x video encoders. Considering that, the most recent international standard, H.264 codec, includes several new algorithms in order to increase compression efficiency and application flexibility. Unfortunately this improved technology demands very high performance to support real-time video encoding. Considering this scenario, the current paper proposes the adoption of the GPU NVidia CUDA® technology as development platform to increase performance of a H.264 intra video encoder. Practical results comparing the proposed solution with the original H.264/AVC reference code points to significant performance gains for high definition computing.

Categories and Subject Descriptors

D.1.3 [**Concurrent Programming**]: Parallel programming

General Terms

Algorithms, Performance, Design.

Keywords

H.264 encoding, GPU, GPGPU, parallel programming.

1. INTRODUÇÃO

A tecnologia do vídeo digital tem levado ao desenvolvimento crescente de aplicações que explorem o uso de recursos de multimídia (web seminários, IPTV, teleconferência, entre outros). Neste contexto, o codificador de vídeo é composto por um conjunto de algoritmos responsáveis por comprimir os dados de vídeo. A compressão de dados se torna parte essencial do processo, pois somente assim pode-se garantir a transferência de vídeos sobre canais de comunicação com banda limitada [1].

Os codificadores de vídeo mais empregados em sistemas comerciais de multimídia (DVDs) e TV Digital (ATSC, ISDB e DVB), utilizam a especificação MPEG-2/H.262. Apesar da grande difusão, esta tecnologia já vem sendo considerada obsoleta, devido às diversas melhorias apresentadas pelo codificador de vídeo H.264, o qual foi inclusive escolhido, em 2005, para ser adotado pelo sistema brasileiro de TV digital (SBTVD). A escolha do H.264 se deve principalmente à maior eficiência de compressão deste algoritmo. Estudos comparativos mostram que para uma mesma qualidade de vídeo, a compressão do algoritmo H.264 ocupa em média 50% da banda ocupada pela compressão pelo algoritmo MPEG-2. Esta característica por si só torna o codificador H.264 mais adequado para redes de banda restrita ou variável, como acontece no caso da teleconferência [1].

Apesar destas vantagens, o desenvolvimento de um codificador de vídeo H.264 em tempo real é ainda um grande desafio devido à complexidade dos algoritmos inovadores deste padrão, o que exige uma capacidade de processamento superior. Para se ter uma idéia desta complexidade, pode-se considerar os resultados de [2] que apontam que um codificador de vídeo H.264, para operar sobre vídeos de alta definição pode exigir a demanda de até 3,6 TIPS (*Tera Instructions Per Second*). Para atender a este elevado desempenho, diferentes soluções em sido propostas nos últimos anos, empregando plataformas com múltiplos microprocessadores de DSP, clusters de computadores, plataformas baseadas em chips dedicados tipo ASIC (*Application Specific Integrated Circuit*) ou então pelo uso de lógicas programáveis. São soluções altamente paralelizáveis, porém durante seu projeto é necessário observar uma série de questões de sincronismo, sinalização, uso de recursos especiais como multiplicadores, bem como sinais de arbitração para evitar problemas no compartilhamento de recursos.

Além destas soluções citadas, tem se mostrado especialmente promissora a última geração de processadores gráficos (*GPU – Graphical Processing Unit*), caracterizados por arquiteturas compostas simultaneamente por múltiplos núcleos de alto desempenho [3]. Considerando este cenário, o presente artigo apresenta uma análise investigativa do uso da tecnologia de GPU NVIDIA CUDA® como plataforma de desenvolvimento para um codificador intra de vídeo H.264.

WebMedia'11: Proceedings of the 17th Brazilian Symposium on Multimedia and the Web. Short Papers.
October 3 -6, 2011, Florianópolis, SC, Brazil.
ISSN 2175-9650.
SBC - Brazilian Computer Society

Como estratégia de validação da proposta, foi utilizado, como referência, o software H.264 da entidade JVT (*Joint Video Team*) [4]. Os resultados obtidos comparando-se o software de referência com a nova arquitetura proposta apontam ganhos de desempenho bastante relevantes.

O artigo é organizado da seguinte maneira: a seção 2 apresenta a arquitetura GPU; a seção 3 descreve a nova arquitetura proposta; a seção 4 mostra os dados reais obtidos comparando o método proposta com uma arquitetura tradicional e finalmente a seção 5 apresenta as considerações finais do artigo.

2. VISÃO GERAL DA TECNOLOGIA DE GPU NVIDIA CUDA®

Nos últimos anos, as demandas crescentes por processamento tridimensional de placas aceleradoras gráficas, principalmente influenciadas pelo mercado de jogos de computador, têm levado a um avanço sem precedentes das arquiteturas de processadores gráficos dedicados. Mas não é somente o mercado de jogos que pode-se valer desta tecnologia. De fato, já se torna comum o conceito de GPGPU (General Purpose Graphical Processing Unit), que promove o uso da tecnologia de processadores gráficos para diferentes aplicações de propósito geral, sendo potencialmente vantajoso para algoritmos que possibilitem tratamento vetorial e paralelizado, explorando o uso de múltiplas tarefas simultâneas [3].

Uma das mais difundidas tecnologias de processadores gráficos é a tecnologia de GPUs da empresa NVIDIA, que incorpora, em um mesmo encapsulamento, processadores de vértice (vertex), de pixel e textura (*pixel shader*), os quais podem operar de forma cooperativa ou independente. De forma geral, as principais entidades destes processadores gráficos são os núcleos de processamento de textura, também chamados de TPC (*Texture Processing Cluster*). Cada um destes núcleos é composto por um processador de textura de ponto flutuante (TEX) e de outros multiprocessadores genéricos, chamados de SM (*Streaming Multiprocessor*). Cada SM, por sua vez, é composto por algumas unidades de função especial (SFU – *Special Function Unit*), responsáveis por operações transcendentais como seno, cosseno, tangente e de diversos processadores genéricos, chamados de SP (*Streaming Processor*). Cada multiprocessador adota internamente uma arquitetura altamente simétrica, que foi idealizada para permitir o acesso simultâneo a memórias e registradores por diversas tarefas paralelas [5].

A comunicação com o mundo externo ocorre por memórias globais com gerência de *cache* dedicada para instruções e dados. Dentro desta arquitetura, podem ser utilizados quatro tipos de memórias de acesso rápido: (i) memória de registradores internos de cada SP; (ii) memória compartilhada (*Shared Memory*) entre SPs de um mesmo SM; (iii) memória de constantes (somente leitura) e (iv) e memória de textura (também somente leitura). Os acessos às memórias de constantes e de texturas são mais rápidos, pois estas possuem memórias *cache* incorporadas que podem ser usadas quando múltiplas *threads* acessam os mesmos valores ou valores adjacentes. Uma representação simplificada da arquitetura de GPUs NVIDIA é mostrada na Figura 1.

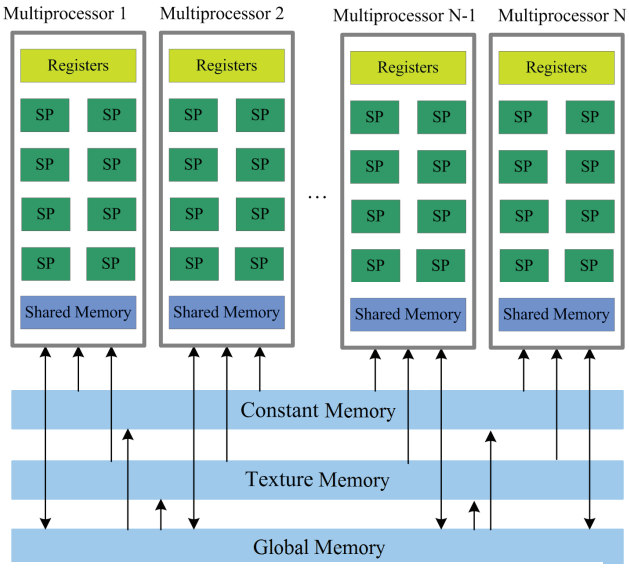

Figura 1. Arquitetura genérica GPU NVIDIA.

Apesar da complexidade desta tecnologia, a idéia é que a exploração de seus recursos ocorra no nível do software e não do hardware. Para facilitar isto, a empresa distribui o *toolkit* CUDA® (*Compute Unified Device Architecture*), que promove a exploração do conceito de programação paralela, em uma metodologia chamada de SPMD (*Single Programming Multiple Data*), que visa aumentar a usabilidade e facilitar o particionamento e sincronismo de algoritmos multi-tarefas [5].

3. SOLUÇÃO PROPOSTA

3.1 Algoritmo de Computação Intra

A arquitetura interna de um codificador H.264 completo é bastante complexa, dados os diferentes algoritmos envolvidos (preditor de movimento, módulo computacional, compensação de movimento, entropia, bloco de filtragem, entre outros). Dentre estes, um dos algoritmos mais importantes é o módulo computacional que é responsável por explorar a redundância espacial dentro de imagens (bloco em destaque na Figura 2).

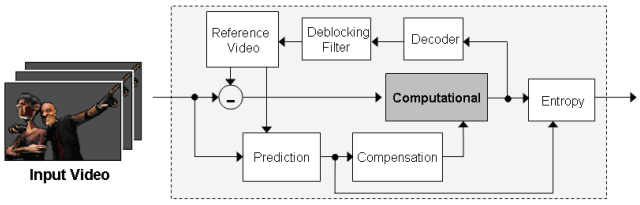

Figura 2. Diagrama de blocos de um codificador H.264.

Na prática, o módulo computacional realiza internamente os cálculos de transformadas e quantização do codificador, os quais são descritos com maiores detalhes na seção a seguir.

3.1.1 Cálculo das transformadas

São dois os algoritmos de transformadas adotados pelo codificador H.264: transformada discreta de cossenos (DCT – *Discrete Cosine Transform*) e Hadamard.

O algoritmo de transformadas DCT, adotado pelo codificador H.264 SVC, deve ser aplicado, a cada vez, sobre blocos de 4x4 pixels do vídeo de entrada ou sobre blocos de 4x4 pixels de resíduos calculados pelo mecanismo de compensação de movimento (Figura 2).

Já o módulo de transformada Hadamard opera apenas sobre os coeficientes DC resultantes da transformada DCT. Quando operando sobre blocos de luminância cada macrobloco de entrada (16x16 elementos) gera 16 componentes DC (decorrente de 16 blocos de 4x4 pixels), que são usados para compor uma matriz de 4x4 elementos DC. Assim o algoritmo interno fica bem parecido com o da transformada DCT. Simetricamente, quando os blocos de entrada são de crominância são gerados blocos de 2x2 elementos DC.

Baseado nestas considerações a implementação de um algoritmo de transformada qualquer, dentro de um codificador H.264, pode ser implementado em duas etapas, conforme identificado a seguir:

Estágio 1: $Y_1 = A.X$ (1)
Estágio 2: $Y = Y_1 .A^T$ (2)

Onde:
X representa o bloco de entrada;
A representa a matriz de transformada;
Y_1 representa o resultado do primeiro estágio;
A^T representa a matriz de transformada transposta.

As matrizes de transformadas do codificador H.264, seja para transformadas DCT ou Hadamard são composta por números inteiros, o que facilita a sua implementação utilizando-se assim apenas aritmética de inteiros [1].

3.1.2 Cálculo da quantização

O algoritmo de quantização do codificador H.264 é usado para reduzir a resolução dos valores de saída de acordo com o parâmetro QP (*Quantization Parameter*), que pode variar de 0 a 51, permitindo assim grande flexibilidade no controle de compressão de dados. De forma geral, o procedimento para implementação deste módulo é definido como [2]:

$$Q(i,j) = (|Y(i,j)|.MF + f) >> qbits$$
$$\text{sign}(Q(i,j)) = \text{sign}(Y(i,j))$$

onde:
$Y(i,j)$ representa o elemento da posição (i, j);
MF é o fator de multiplicação;
f é um valor escolhido de arredondamento;
$qbits$ representa uma divisão por deslocamento de bit.

Para fins de aumento de simplificação do algoritmo os valores de MF, f e $qbits$ podem ser obtidos somente consultando tabelas próprias indexadas pelo valor de QP e a posição (i, j).

3.2 Exploração da Arquitetura GPU CUDA®

Os programas compilados para a arquitetura CUDA são organizados em blocos de tarefas (*thread blocks*) que podem ser executadas em paralelo dentro de cada SP, dependendo do algoritmo a ser adotado e dos recursos utilizados (memória e registradores compartilhados). Para tanto se buscou explorar a simetria de dados destes algoritmos para aumentar a eficiência do SPMD e por conseqüência o paralelismo efetivo dentro da plataforma [6]. O conjunto de tarefas que executam em paralelo dentro de um mesmo SM é chamado de *warp*. Segundo dados de [5] o número máximo de tarefas simultâneas a se obter por SM é 32. Assim, optou-se por alinhar o algoritmo proposto com blocos de entrada de 16x16 elementos, tanto nas transformadas como na quantização. Este alinhamento possibilita um processamento de até duas linhas de pixels por SM (Figura 3).

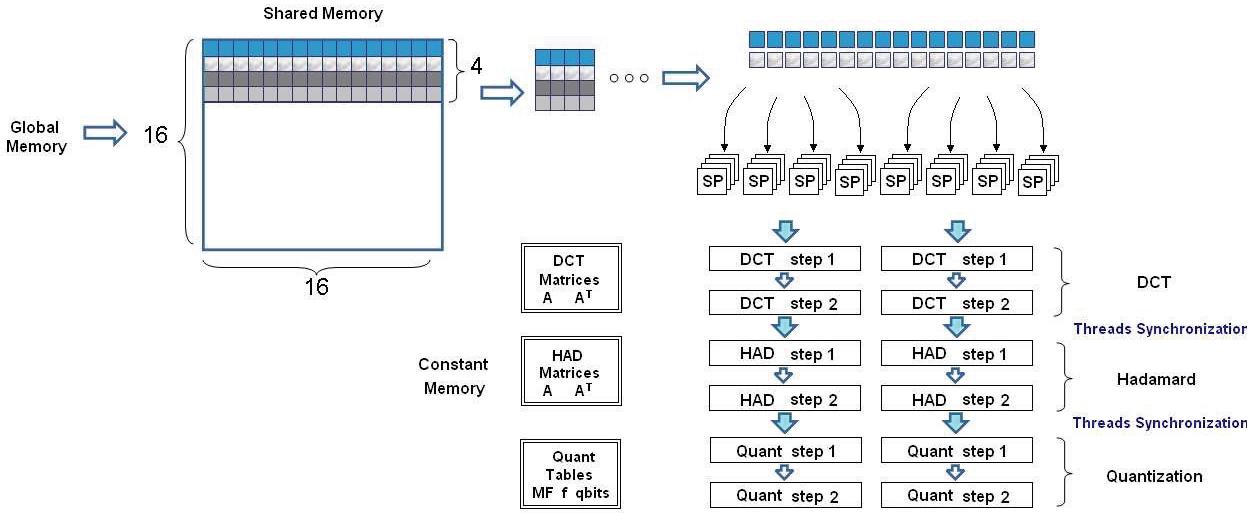

Figure 3. Arquitetura proposta para módulo computacional usando GPUs Nvidia.

Além do alinhamento de dados para propiciar maior paralelismo, a proposta também observou os diferentes tipos de memórias disponibilizadas pela plataforma. Cada vez que uma transferência de dados é realizada, estes são armazenados na memória global, que é significativamente mais lenta que as memórias internas do SM. Assim sendo a abordagem foi de fazer o menor número de acessos à memória global, que foi usada apenas como interfaces de entrada e saída de dados da placa.

As transferências de dados ocorrem com pacotes que comportam múltiplos de macroblocos H.264 (16x16 pixels), de forma a reduzir as latências de comunicação entre CPU e GPU (latências estas fortemente dependentes do sistema operacional) [7].

É importante notar adicionalmente que as diferentes memórias da arquitetura NVIDIA possuem diferentes tempos de acesso.

Neste sentido a memória de constante foi preenchida com as matrizes e tabelas citadas nos algoritmos enquanto que a memória compartilhada foi utilizada para o salvamento dos dados processados em si. Observando-se a Figura 3 esta arquitetura pode ser melhor entendida. Todos os processos foram implementados em duas etapas, sendo utilizadas funções de sincronismo entre processos para garantir consistência dos dados [8].

4. RESULTADOS EXPERIMENTAIS

Conforme já comentado para a validação da proposta esta foi comparada com o software de referência H.264 versão 13.0, que é disponibilizado pela entidade JVT, como forma de auxiliar pesquisadores no entendimento deste padrão de codificação [7].

Como metodologia de comparação o código de referência foi adaptado para duas versões, uma onde utiliza somente uma CPU Intel QuadCore 2,8 GHz e outra onde o módulo computacional é calculado por uma placa gráfica externa, que emprega a tecnologia de GPU. Duas placas gráficas foram utilizadas nestes experimentos: (i) Placa GeForce 8500GT, que adota uma arquitetura com 8 SMs rodando a 450 MHz e (ii) GeForce GTX560Ti, que possui 384 SMs, operando a uma freqüência de 822 MHz.

Dois cenários foram avaliados, o primeiro considerando-se vídeos em resolução 4CIF (704x576) e o segundo considerando HD (1920x1080 pixels). As tabelas a seguir resumem os resultados de tempos médios por quadros em unidades de milissegundos.

Tabela 1. Comparação dos tempos de processamento 4CIF

	Configuração		
	CPU	8500GT	GTX560Ti
Computacional (Transformadas + Quantização)	**40,99 ms**	**70,75 ms**	**11,27 ms**
Ganho obtido	**1**	**0,58**	**3,63**

Tabela 2. Comparação dos tempos de processamento HD

	Configuração		
	CPU	8500GT	GTX560Ti
Computacional (Transformadas + Quantização)	**208,71 ms**	**303,57 ms**	**53,89 ms**
Ganho obtido	**1**	**0,81**	**3,87**

Os resultados mostram que placas com pequeno número de multiprocessadores (8500GT), mesmo desempenhando múltiplas tarefas simultâneas, tem seu desempenho atenuado devido a sua reduzida freqüência de trabalho comparada com CPUs de computador. Já quando a GPU possui um número elevado de multiprocessadores (GTX560Ti) o número de tarefas paralelas aumenta proporcionalmente aprimorando seu desempenho global.

5. CONCLUSÕES

A solução proposta, que apresenta uma solução codificação intra H.264, utilizando a tecnologia de GPUs NIVIA CUDA®, se mostra muito promissora, principalmente quando elevados níveis de paralelismo são possíveis. Experimentos práticos indicam que para vídeos de resolução SD, atinge-se um ganho de desempenho de 3,6 vezes, enquanto que para vídeos maiores o ganho passa de 3,8 vezes.

6. REFERÊNCIAS

[1] Richardson I. E. G.; "H.264 and MPEG-4 Video Compression". England, Ed. Wiley & Sons, v2. 2003, 281p.

[2] Chen, and H.-Y. Chen, "Physical Design for System-On-a-Chip" in Essential Issues in SOC Design (Y.-L. Lin, Editor), Springer, 2009.

[3] Do Dinh M. T. GPUs - Graphics Processing Units. Institute of Computer Science, University of Innsbruck. July 7, 2008.

[4] ITU, "H.264/AVC Reference Software Decoder (version 13.0)". 2008. <http://iphome.hhi.de/suehring/tml/doc/ldec/html.

[5] W. -N. Chen, H. -M. Hang, "H.264/AVC motion estimation implementation on Compute Unified Device Architecture (CUDA)", IEEE ICME-08, pp. 697-700, 2008.

[6] Ren j. et al. Parallel Streaming Intra Prediction for Full HD H.264 Encoding IEEE ICME-10. pp. 978-983, 2010.

[7] Huang Y.-L., Shen Y.-C., Wu J.-L.Scalable computation for spatially scalable video coding using NVIDIA CUDA and multi-core CPU. MM'09, October, 2009, China. pp.361-370.

[8] Jiao L., Zhou J., Chen R. Efficient Parallel Intra-prediction Mode Selection Scheme for 4×4 Blocks in H.264. IEEE ICICTA 2010, China, April 2011, p. 527-530.