IMPLEMENTAÇÃO MULTITHREADED DO MÓDULO DE COMPENSAÇÃO DE MOVIMENTO DO PADRÃO H.264/AVC UTILIZANDO PROGRAMAÇÃO DE PROPÓSITO GERAL EM UNIDADE DE PROCESSAMENTO GRÁFICO

GNUTZMANN, Pamela P.[1]; ATAÍDES, Vitor A.[1]; FERNANDES, Paulo E. F.[1]; AGOSTINI, Luciano V.[2]; PILLA, Maurício L.[2]

[1]UFPel, Centro de Desenvolvimento Tecnológico.
Email: {ppgnutzmann, vaataides, peffernandes}@inf.ufpel.edu.br
[2]UFPel, Centro de Desenvolvimento Tecnológico.
Email: {agostini, pilla}@inf.ufpel.edu.br

1 INTRODUÇÃO

Atualmente, observa-se uma redução do crescimento do desempenho dos processadores de uso geral, devido a limitações físicas dos chips. Essas limitações são contornadas com a inclusão de mais processadores em um único chip, os chamados multicores (ASANOVIC, 2009).

As GPUs (*Graphics Processing Units*) exploram paralelismo há bastante tempo, usando uma arquitetura SIMD (*Single Instruction Multiple Data*), hoje também denominada SIMT (*Single Instruction Multiple Threads*). Uma GPU, hoje em dia pode ter mais de 350 núcleos e cada um executa até 8 threads (processos leves) ao mesmo tempo (NVIDIA Corporation 2010).

GPGPU (*General-Purpose on Graphics Processing Units*) é a técnica de usar a GPU, normalmente usada para processamento de computação gráfica, para realizar a computação de aplicações que normalmente são feitas pelo processador principal do computador. O uso da GPU pode ser bastante eficiente, se usado devidamente, pelo seu grande número de processadores. Conforme pode ser observado na Figura 1, esse grande número de *threads* faz com que o desempenho das placas gráficas ultrapasse em muito o desempenho das CPUs para operações de ponto flutuante com precisão simples (NVIDIA Corporation 2010).

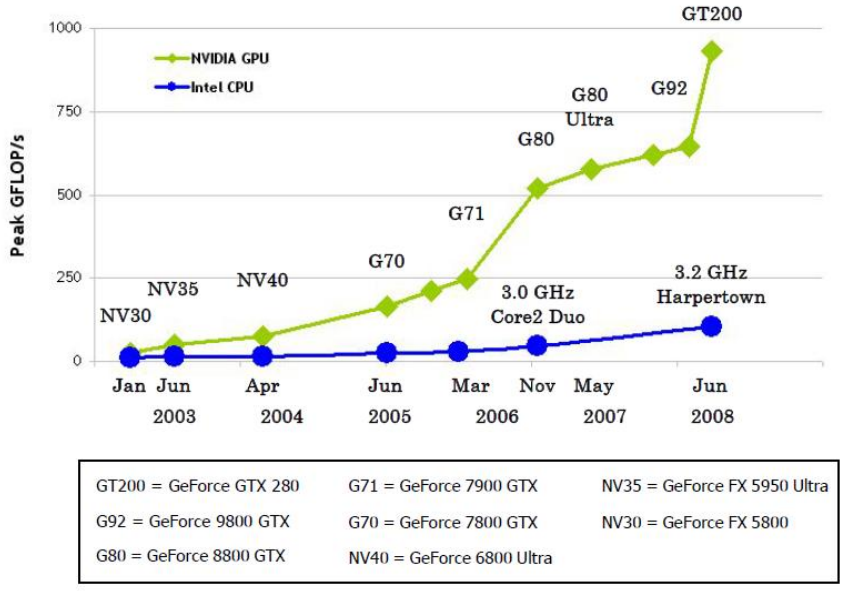

Figura 1 – Comparativo entre desempenho máximo entre CPUs e GPUs (NVIDIA, 2009)

É facilmente perceptível, baseado no gráfico, que o desenvolvimento em GPGPU tende a ser uma alternativa cada vez mais explorada nos próximos anos, especialmente para aplicações de elevada complexidade computacional, como a codificação e decodificação de vídeo. De fato, enquanto os processadores vêm diminuindo seu ritmo de crescimento de desempenho, a qualidade e a resolução dos vídeos digitais não para de crescer. Por exemplo, os vídeos FULL HD (*High Definition*), com resolução de 1920x1080 *pixels* estão presentes até nos celulares e *smartphones*. Armazenar ou transmitir todos os *pixels* dos quadros de um vídeo sem compressão ocuparia uma quantidade exorbitante de recursos. Deste modo, a eficiência da codificação e decodificação de vídeo é essencial para o sucesso nas aplicações que manipulam vídeos de alta resolução.

Este trabalho tem por objetivo utilizar o alto poder de paralelismo da GPU para executar parte do algoritmo do módulo de Compensação de Movimento (MC), presente no decodificador de vídeo do padrão H.264.

Uma das etapas da codificação de vídeo consiste na busca de redundâncias entre *frames* (quadros) de um vídeo. Para essa busca, os *frames* são divididos em blocos e é feita uma comparação entre os blocos do quadro de referência e do atual, que está sendo codificado. Porém, nem sempre os blocos mais parecidos encontram-se em posições equivalentes nos dois quadros. Para indicar o deslocamento dos blocos similares do quadro de referência para o atual, são gerados os vetores de movimento, que são pares de inteiros <x,y> representando o deslocamento de cada bloco. Para cada quadro codificado, é gerada uma matriz destes vetores.

A função do módulo de Compensação de Movimento é realizar, durante a decodificação de um vídeo, a reconstrução de cada quadro de acordo com seus respectivos quadros de referência e matrizes de vetores, copiando os blocos pro quadro a ser construído (AGOSTINI, 2007), conforme pode ser observado no exemplo na Figura 2. Nos Quadros de Referência e Construído , a numeração serve para identificar os blocos. Na Matriz de vetores de Movimento, os pares de inteiros indicam o deslocamento que os blocos do Quadro de Referência devem sofrer para formarem o Quadro Construído pela MC. O primeiro inteiro indica o deslocamento horizontal do bloco, e o segundo, o deslocamento vertical.

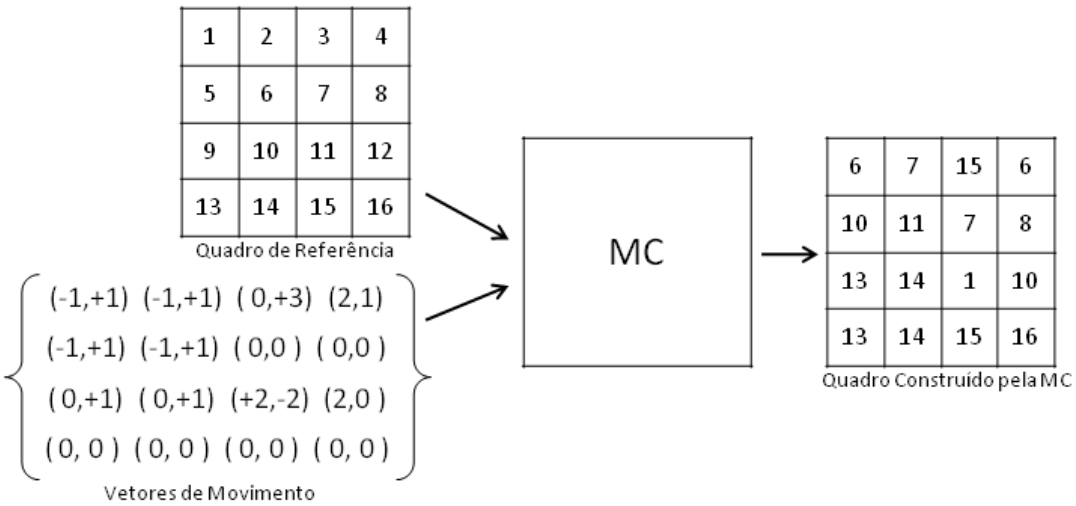

Figura 2 – Esquema de entradas e saídas do módulo da MC

2 METODOLOGIA (MATERIAL E MÉTODOS)

Ao realizar um estudo sobre o módulo de Estimação de Movimento, foi possível perceber seu grande potencial de paralelismo, visto que a compensação de movimento realizada para cada bloco de um quadro é independente. Para paralelizar os cálculos da MC utilizando a GPU, foi utilizada a arquitetura CUDA(*Compute Unified Device Architecture*), das novas placas da NVIDIA (NVIDIA Corporation, 2008). Na construção do programa foi utilizada a biblioteca CUDA para a linguagem C, que facilita a programação em placas gráficas.

Para gerar as entradas do algoritmo proposto, foi utilizado um algoritmo sequencial que recebe como entrada um vídeo e realiza os cálculos de estimação de movimento para cada um de seus *frames*, retornando como saída uma sequência de matrizes contendo vetores de movimento e os quadros de referência utilizados para gerar cada matriz de vetores.

Cada matriz corresponde a um quadro do vídeo, e cada vetor indica o deslocamento de um bloco do quadro. Após a leitura de cada matriz, 32400 fluxos de execução são disparados ao mesmo tempo, cada um ficando responsável por realizar a compensação para um vetor de movimento. Então, as *threads* realizam a cópia do bloco correspondente no quadro de referência para o quadro sendo construído, levando em conta o movimento indicado pelo vetor. Em seguida, outro *frame* do vídeo é carregado da memória e processo é repetido, continuando assim até o final do vídeo.

3 RESULTADOS E DISCUSSÃO

Após o estudo do funcionamento da Estimação de Movimento, o algoritmo paralelizado foi implementado e diversos testes foram feitos, utilizando a GPU NVIDIA GeForce GTX285, que possui 240 núcleos.

O programa foi executado para vídeos de 200 *frames* com a resolução 1920x1080 *pixels*. Cada *frame* foi dividido em blocos 8x8. Testes utilizando as mesmas configurações foram realizados também para uma versão sequencial do algoritmo. Os testes foram repetidos 10 vezes cada um, para a obtenção de resultados mais confiáveis.

O tempo de execução foi monitorado e a versão paralelizada do algoritmo teve seu tempo de execução aproximadamente 2,1 vezes menor do que a versão sequencial.

É importante perceber que, apesar do uso de 240 núcleos de processamento, o tempo não diminui em 240 vezes. Isso ocorre pois a frequência de um único processador da GPU é inferior ao de uma CPU. Notemos que, se o mesmo vídeo for processado por um único núcleo da GPU, então o tempo de execução será aproximadamente 240 vezes maior que o de uma execução com 240 núcleos.

4 CONCLUSÃO

A implementação do algoritmo de Compensação de Movimento em GPU foi concretizada e o objetivo de demonstrar as potencialidades da exploração do paralelismo para Compensação de Movimento foi alcançado. Os resultados indicam uma aceleração de 2,1 vezes na execução da MC com a solução desenvolvida neste trabalho, o que é um ganho muito expressivo.

Como trabalho futuro pretende-se estudar o novo padrão de compressão de vídeo HEVC e paralelizar módulos do codificador e decodificador do padrão utilizando GPGPU. Também pretende-se buscar otimizações para o módulo de Compensação de Movimento descrito nesse trabalho.

5 REFERÊNCIAS

NVIDIA CORPORATION. **NVIDIA CUDA: Programming Guide Version 3.0**. Santa Clara, EUA: NVIDIA Corporation, 2008. 98 p.

ASANOVIC, K.; BODIK, R.; DEMMEL, J.; KEAVENY, T.; KEUTZER, K.; KUBIATOWICZ, J.; MORGAN, N.; PATTERSSON, D.; SEN, K.; WAWZRYNEK, J.; WESSEL, D.; YELICK, K. **A view of the parallel computing landscape**. Communications of the ACM, 52(10):56-67, 2009

AGOSTINI, L. V. . **Desenvolvimento de Arquiteturas de Alto Desempenho Dedicadas à Compressão de Vídeo Segundo o Padrão H.264/AVC**. Ago 2007. Tese de Doutorado em Ciência da Computação - Ufgrs, Porto Alegre.