A PLATAFORMA DE PROGRAMAÇÃO OPENCL COM A ARQUITETURA CUDA

Thiago Sousa Santos (bolsista ICV), Kelson Rômulo Teixeira Aires (Orientador, Departamento de Informática e Estatística – UFPI)

1 Introdução

A visão computacional tem por objetivo elaborar uma visão artificial, por meio de técnicas de processamento de imagens, com a finalidade de tornar o computador capaz de perceber e analisar o meio no qual se encontra. Uma imagem digital é formada por um conjunto de pixels. A resolução de uma imagem define a quantidade de pixels existentes na horizontal e vertical respectivamente. Os algoritmos utilizados no processamento de imagens são repetitivos em função da quantidade de pixel a serem computados, e quanto maior a resolução de imagem utilizada maior a carga de dados a ser analisada, e consequentemente maior a demanda de processamento por tempo.

A programação paralela é um termo utilizado para definir um processamento simultâneo de dados em vários núcleos. Uma forma de realizar a programação paralela é através de um computador principal que divide a execução de um algoritmo entre outros. Essa técnica por sua vez requer um custo elevado em função do número de computadores utilizados.

O uso de unidades gráficas de processamento (GPU - *Graphics Processing Units*) é outra forma de programação paralela. As placas gráficas são utilizadas em maioria como aceleradores de vídeo, limitando o seu verdadeiro potencial. A tecnologia *multi-nuclear*, existente nas GPUs, não é restrita a renderizar imagem. As *threads* podem executar, de forma independente, algoritmos simultaneamente. Essa característica se mostra eficiente na execução de algoritmos repetitivos, nos quais os cálculos são independentes.

2 Metodologia

As etapas foram executadas sequencialmente e dentro dos prazos estipulados, pra que o projeto não atrasasse. As etapas começaram pelo levantamento de bibliografia das ferramentas que foram utilizadas projeto, seguido de um estudo e compreensão das mesmas. Desta forma, foi desenvolvida uma base de conhecimento utilizado para integrar e programar algoritmos nas ferramentas.

Na primeira etapa foi feita um levantamento da bibliografia da arquitetura CUDA, entendendo seu funcionamento e buscando trabalhos relacionados à mesma. Ao final foi feito uma análise do material elaborado e uma viabilização da integração da plataforma OpenCL e com a CUDA.

Na segunda etapa, um estudo dos algoritmos e técnicas utilizando CUDA, envolvendo processamento de imagens e visão computacional, foi feito e algoritmos como detectores de bordas foram escolhidos para serem implementados em OpenCL com CUDA. Na etapa que se segue, os algoritmos escolhidos foram implementados e testados. Algoritmos esses, com alta carga de dados a serem processados e com bloco de códigos repetitivos e independentes.

Na ultima etapa do projeto, um artigo foi elaborado. Este artigo é a síntese da analise dos resultados obtidos na visão computacional utilizando o OpenCL com CUDA em comparação a tecnologias convencionais. Todo o material produzido ao longo do projeto também está disponibilizado neste artigo.

3 Resultados e Discussão

3.1 CUDA

A NVIDIA, fabricante de unidades de processamento gráfico, desenvolveu a tecnologia CUDA (*Compute Unified Device Architecture*), uma arquitetura de programação paralela com objetivo de viabilizar, de forma simples, a programação paralela em suas GPUs. Essa tecnologia permite a utilização dos núcleos da GPU, de forma simultânea e independente, para a execução de blocos de códigos. A arquitetura CUDA diminui o custo de hardwares na aplicação da programação paralela no seu estudo ou na elaboração de softwares científicos e comerciais.

A tecnologia CUDA é uma proposta geral de arquitetura de computação paralela da NVIDIA. Em novembro de 2006, a NVIDIA introduziu CUDA como um novo modelo de programação paralela e *Instruction Set Architecture* (ISA) [1]. CUDA aproveita o mecanismo de computação paralela em GPUs NVIDIA para resolver muitos problemas computacionais complexos de maneira mais eficiente do que em uma CPU. CUDA disponibiliza um software que permite aos desenvolvedores usar o C como uma linguagem de programação de alto nível, outras linguagens ou interfaces de programação de aplicativos são suportadas, tais como FORTRAN CUDA, OpenCL, Phyton e computação direta.

3.2 OpenCL

O framework OpenCL (*Open Computing Language*) foi proposto pela Apple para padronizar a programação em GPUs. O OpenCL é uma ferramenta de programação paralela compatível com diferentes sistemas operacionais e diversos dispositivos. Dessa forma o OpenCL torna o algoritmo paralelo portável a plataforma utilizada.

As rotinas iniciais de detecção do sistema operacional e dispositivos moldam o método de trabalho do OpenCL. Tais rotinas ajustam-no ao sistema tal que um mesmo software possa ter diferentes implementações para cada máquina que por ventura execute o programa, o contexto.

As etapas seguintes geram o código através de implementações das funções em OpenCL C. Devido à memória dos dispositivos não permitir acesso direto, são usados funções da ferramenta para alocação, escrita e leitura dos dados e após o uso ocorre à liberação dos mesmos. O uso das implementações do programador é indireto utilizando funções padrões para passagem de parâmetros e execução paralela.

A partir dos dados obtidos nas rotinas, o programa é compilado em tempo de execução, adequando o código OpenCL C. Uma variante da linguagem C o que o torna amigável, facilitando seu estudo. Há o acréscimo de novos tipos em relação a ISO C99 como os específicos para processamento de imagem e o booleano incorporado pela linguagem C++ e vetores de tamanhos variados para acelerar o processo de transferência [2, 3].

4 Resultados

O CUDA e o OpenCL oferecem duas interfaces para programar GPUs. O OpenCL é uma interface padrão aberta que permite programar CPUs, GPUs e outros dispositivos de diferentes empresas, enquanto o CUDA é especifico das GPUs NVIDIA. Entretanto OpenCL promete uma portabilidade de linguagem para programar em GPU. Essa generalidade implica em uma perda de desempenho. As duas interfaces de programação possuem funcionalidades similares, no entanto o desempenho do kernel do OpenCL é entre 13% e 64% mais lento que CUDA [4].

No projeto, algoritmos como a detecção de uma borda já foram implementados em GPU tanto em CUDA como na integração OpenCL mais CUDA. Uma borda em uma imagem possui uma característica relevante, como uma mudança no nível de intensidades dos pixels. Mecanismos detectores de bordas são elaborados a fim de encontrar pixels onde ocorre esse tipo de variação. Dessa forma quando esses pixels estão próximos podem ser destacados formando uma borda ou contorno [5].

Para a elaboração dos resultados dos detectores de bordas, imagens com diferentes dimensões foram utilizadas. Para o filtro de Sobel, utilizando imagens de resoluções inferiores a 400x300 *pixels*, os resultados obtidos com processamento em GPUs não foram satisfatórios por executarem o algoritmo mais lento que em CPU. Para imagens com resoluções superiores a 400x300, a GPU é mais veloz que a CPU. Nos experimentos, o processamento via GPU chegou a ser 2.83 vezes mais rápido do que em CPU.

5 Conclusão

Na visão computacional, o uso das GPUs se mostrou muito eficiente. Processamento de imagens requer alto poder computacional, principalmente quando há necessidade de processar os dados em tempo-real. Em imagens de alta resolução, a execução paralela de cálculos de derivadas e K-Means em GPUs possibilita a obtenção de resultados em tempo-real. Isso se explica pelo fato destes algoritmos serem amplamente paralelos, ou seja, cada cálculo pouco depende dos anteriores. O processamento em tempo-real pode ser utilizado aplicações onde informações são retiradas de um *stream* de vídeo, e neste caso é necessário computar os frames do vídeo. Se o processo for lento, haverá perda ou inconsistência da informação.

Neste trabalho abordamos algumas técnicas utilizadas na visão computacional e suas aplicações. Também foi abordado, duas fortes ferramentas para programação paralela em GPUs, o CUDA e OpenCL tentando explicar o funcionamento delas. Buscamos demonstrar a contribuição das GPUs na visão computacional, bem como seu desempenho devido à paralelização dos algoritmos utilizados. E assim contribuir para a continuidade e o desenvolvimento dos algoritmos aqui expostos, divulgando a programação paralela em GPUs e o seu uso na visão computacional.

Referências Bibliográficas

[1] NVIDIA, *OpenCL Programming Guide for the CUDA Architecture Version 2.3*. NVIDIA Corporation: Santa Clara, Califórnia, 2010.

[2] http://www.khronos.org/opencl/.

[3] KHRONOS, *Introduction and Overview*. Khronos Group, June 2010.

[4] KARIMI, Kamran; DICKSON, Neil G.; HAMZE, Firas. *A Performance Comparison of CUDA e OpenCL*. D-Wave Systems Inc.: Canada, 2010.

[5] MARENGONI, Maurício; STRINGHINI, Denise. *Introdução à Visão Computacional usando OpenCV*. RITA 2009.

Palavras-chave: Visão computacional, Programação paralela, GPU, CUDA, OpenCL.