CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Jogos Digitais

ROBERTO NOVAKOSKY

COMPUTAÇÃO ATRAVÉS DA GPU

ÊNFASE EM JOGOS DIGITAIS

Americana, SP

2014

CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Jogos Digitais

ROBERTO NOVAKOSKY

COMPUTAÇÃO ATRAVÉS DA GPU

ÊNFASE EM JOGOS DIGITAIS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do curso de Tecnologia em Jogos Digitais, sob a orientação do Prof. Me. Francesco Artur Perrotti.

Área de Concentração: Jogos Digitais.

Americana, SP

2014

ROBERTO NOVAKOSKY

COMPUTAÇÃO ATRAVÉS DA GPU

ÊNFASE EM JOGOS DIGITAIS

Trabalho de Graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Desenvolvimento de Jogos Digitais pelo CEETEPS/ Faculdade de Tecnologia - FATEC Americana.

Área de Concentração: Jogos Digitais.

Americana, 24 de junho de 2014.

Banca Examinadora:

_____
Francesco Artur Perrotti (Presidente)
Mestre
FATEC AMERICANA

_____
Wagner Siqueira Cavalcante (Membro)
Mestre
FATEC AMERICANA

_____
Benedito Aparecido Cruz (Membro)
Especialista
FATEC AMERICANA

Agradeço em especial a meu professor orientador, Francesco Artur Perrotti, que me ajudou para a conclusão deste trabalho e a minha família pelo apoio e paciência dos dias de pesquisa e envolvimento com a Tecnologia da Informação.

RESUMO

O presente trabalho visa explicar o que é a GPGPU (Computação de Propósito Geral através de GPU – *Graphics Processing Unit*), mostrar sua relação com os profissionais de Jogos Digitais e como escolher adequadamente soluções convencionais ou por GPGPU, dependendo do problema que se apresenta. Envolvendo o assunto, também são abordados alguns conceitos relevantes de APIs (*Application Programming Interface*) - Interface de Programação de Aplicativos. A respeito da arquitetura de GPU, para se basear este trabalho, foi escolhida a arquitetura presente nas Placas de Vídeo do fabricante NVIDIA, que adota a tecnologia CUDA (*Compute Unified Device Architecture*), uma plataforma de computação paralela de propósito geral, ferramenta de programação de GPGPU em suas placas. Pensando no Profissional de Jogos Digitais, há impacto e implicações em várias linhas de trabalho, como por exemplo, o uso desta tecnologia no desenvolvimento das *engines* de games, para simulações de movimentos da água, ventos, iluminação, simulações destinadas somente a entretenimento ou para aplicações sérias, para a programação da Inteligência Artificial dos personagens, para o reconhecimento de imagens, desenvolvimento de ambientes em Realidade Virtual, em simulação de ambientes para se aproximar ao máximo de um ambiente real, para interações cada vez melhores em tempo real entre a máquina e o jogador.

Palavras Chave: Jogos Digitais; Computação Paralela; GPGPU.

ABSTRACT

This work aims to explain what is GPGPU (General Purpose Computation by GPU - Graphics Processing Unit), show its relationship with the professionals of Digital Games and how to properly choose conventional or GPGPU solutions, depending on the problem at hand. Involving the subject, are also addressed some relevant concepts of APIs (Application Programming Interface). Regarding the GPU architecture, to base this work, was chosen the architecture of NVIDIA Video Cards manufacturer, which adopts CUDA (Compute Unified Device Architecture), a platform for parallel computing, general purpose technology tool GPGPU programming on their boards. Thinking in Professional Digital Games, there are impact and implications in various lines of work, such as the use of this technology in the development of engines for games, for simulations of water movement, wind, lighting, simulations intended only for entertainment or serious applications, for programming the AI of the characters, for the recognition of images, development of virtual reality environments in simulation environments to approximate the maximum of a real environment, for better and better real-time interactions between machine and player.

Keywords: *Digital Games, Paralell Computing, GPGPU.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GPGPU – *General Purpose Computation by* GPU – Computação de Propósito Geral através de GPU.

GPU – *Graphic Processing Unit* – Unidade de Processamento Gráfico.

API – *Application Programming Interface* - Interface de Programação de Aplicativos.

CUDA - *Compute Unified Device Architecture* – É uma plataforma de computação paralela da NVIDIA que tira proveito das GPUs.

GFLOP – *Gigaflop* – equivale a 10^9 (10 seguido de 9 zeros) FLOPS. Unidade de medida geralmente usada em assuntos de performance e desempenho em computação.

TFLOP – *Teraflop* – equivale a 10^{12} (10 seguido de 12 zeros) FLOPS. Unidade de medida geralmente usada em assuntos de performance e desempenho em computação.

FLOPS – *Floating-Point Operations* – Operações de Ponto Flutuante. FLOPS pode significar também Operações de Ponto Flutuante por Segundo dependendo do contexto.

GHz – *Gigahertz* – Equivale a 1 bilhão de hertz (10^9 Hertz). Em computação é uma Unidade de Medida utilizada para indicar a velocidade de um processador.

MHz – *Megahertz* – Equivale a 1 milhão de hertz (10^6 Hertz). Em computação é uma Unidade de Medida utilizada para indicar a velocidade de um processador.

GB – *Gigabyte* – Equivale a 1024^3 bytes.

SURF – *Speeded Up Robust Features* – Algoritmo utilizado em visão computacional para Detecção de Pontos de Interesse.

CPU – *Central Processing Unit* – Unidade Central de Processamento.

TEGRA – Um dos modelos de processadores da NVIDIA para dispositivos móveis.

GTX – Uma das famílias de placas de vídeo da NVIDIA.

SM – *Stream Multiprocessor* – Um bloco existente nos processadores gráficos que possui vários Stream Processors e outros elementos.

SP – *Stream Processor* – Nas placas de vídeo da NVIDIA também é conhecido por CUDA *core*, um dos núcleos dos processadores gráficos.

SETI – *Search for Extraterrestrial Intelligence* – Busca por Inteligência Extraterrestre.

FFT – *Fast Fourier Transform* – Transformadas Rápidas de Fourier.

SDK – *Software Development Kit* – Kit de Desenvolvimento de Software.

NVIDIA - NVIDIA Corporation, empresa fabricante de processadores gráficos.

1 INTRODUÇÃO

GPU – *Graphic Processing Unit* (Unidade de Processamento Gráfico) é um microprocessador especializado em processamento e renderização de imagens. Normalmente esses microprocessadores são encontrados em placas de vídeo, mas também existem versões que podem ser integradas em placas-mãe de computadores (as chamadas placas de vídeo *onboard*). Atualmente também são comuns em celulares, *smartfones*, *tablets* e outros dispositivos com capacidade de processamento gráfico.

As GPUs vêm evoluindo dia a dia, cada versão com maior capacidade de processamento que a anterior. Como seu propósito é o processamento gráfico, são muito utilizadas em jogos de computador, mas já faz algum tempo que profissionais da computação têm procurado aproveitar o poder das GPUs para aplicações mais genéricas que não necessariamente envolvem imagens, o que proporcionou o nascimento da GPGPU – *General-Purpose Computation on GP*U (Computação de Propósito Geral Através de GPU).

Outro fator que incentiva o uso de GPUs é o custo. Um supercomputador do início do ano 2000 custava cerca de US$ 200 milhões e tinha capacidade de 4,9 TFLOP/s (ISTOÉ Dinheiro, 2011). Em 2011, um computador de mesa da NVIDIA com a capacidade de 4 TFLOP/s, custava em torno de R$ 35.000,00 e considerando ainda outras vantagens, manutenção com custo menor, menor consumo de energia e ocupação de espaço físico. A evolução das GPUs fez com que seja possível ter em casa supercomputadores para fins científicos e outros.

Imagem 1

Arnaldo Tavares, da NVIDIA, com um Supercomputador de Mesa

Fonte: ISTOÉ Dinheiro. Um supercomputador de mesa. Ed. Digital 719 de 15/07/2011.

O fato do computador de mesa da foto 1, ser chamado de supercomputador, baseia-se na capacidade de processamento das GPUs nele instaladas e o seu uso através de GPGPU, um uso da GPU que antes não existia.

A GPGPU traz diversos benefícios para a computação, principalmente relacionados com o processamento de imagens nos Jogos Digitais. É uma solução que resolve de forma eficiente vários tipos de problemas que exigem muito processamento ou respostas computacionais em tempo real. Favorece diversos profissionais e cientistas que procuram criar simulações do mundo real, profissionais de propaganda que criam vídeos animados, criadores de jogos, desenvolvedores de *engines* para jogos, pessoas que criam *softwares* de Realidade Aumentada, empresas que criam produtos para fazer reconhecimento facial através de imagens, entre outros.

Como pode ser notado, a GPGPU tem aplicação em diversas áreas, não apenas Jogos Digitais. Muito material tem aparecido recentemente e muitas universidades no mundo têm incluído o tema em seus programas de aprendizagem com seus alunos, mais de 400 universidades ensinam CUDA (NVIDIA Corporation, 2014e). Infelizmente, no Brasil ainda não existe muito material a respeito em português. Um dos objetivos deste trabalho é fazer uma introdução ao assunto para aqueles, que apesar de estudarem ou trabalharem com computação, são ainda leigos sobre os recursos da GPGPU.

O capítulo 2, procura dar uma textualização inicial para um entendimento introdutório, isto é para o leitor compreender melhor como se chegou ao nível tecnológico atual e entender as diferenças entre tecnologias como DirectX®, OpenGL, OpenCL e outras. O capítulo 3, mostra aplicabilidades com a GPGPU - onde ela é usada. No capítulo 4, o leitor terá condições de identificar em qual tipo de problema pode haver uma solução promissora com uma GPU, em que momento pode valer a pena ou não a GPGPU. O capítulo 5, mostra avaliações práticas comparando execução de aplicações em GPU *versus* CPU.

Finalmente espera-se prover um entendimento, pelo menos básico, de como aplicar adequadamente a GPGPU conforme o problema que se apresenta em diversas áreas da computação.

2 A COMPUTAÇÃO COM GPUs

2.1 O QUE É A COMPUTAÇÃO ATRAVÉS DE GPU

Em poucas palavras é a utilização da GPU em conjunto com a CPU para acelerar aplicativos de uso geral nas áreas científica, de engenharia e outras. A Computação através da GPU também é conhecida pela sigla GPGPU - *General-Purpose Computation on GPU* (NVIDIA Corporation, 2014b), cuja tradução pode ser "Computação para fins gerais em dispositivos gráficos".

Basicamente a GPGPU consiste em colocar várias *threads*[1] ao mesmo tempo para processamento em blocos da GPU. Uma GPU pode ter centenas ou até milhares de núcleos de processamento capazes de funcionar em paralelo. Pela quantidade de núcleos que uma GPU pode ter, a computação paralela através dela traz desempenho sem precedentes por causa dos vários blocos sendo executados ao mesmo tempo. O segredo, na verdade, não são somente as *threads*, mas a forma lógica com que são alocados os dados nos blocos.

A combinação CPU e GPU resulta em um sistema muito eficiente. Enquanto a CPU se encarrega de parte das instruções, a GPU usa seus muitos núcleos menores para executar tarefas em paralelo.

2.2 EVOLUÇÃO DA GPGPU

Convencionalmente, GPUs são dispositivos especializados em processamento gráfico. Na chegada do ano 2000, os controladores de vídeo já possuíam especialização para processamento gráfico, fazendo transformação de vértices,

[1] *Thread*: é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas simultaneamente em mais de um núcleo de processador. Pode-se entender para este trabalho como instruções sendo executadas simultaneamente em mais de um núcleo da GPU.

iluminação e alguma textura. Em 1999 a NVIDIA fabricou a placa de vídeo GeForce 256 com capacidade de rasterização e para diferenciá-la utilizou o termo "*Graphics Processing Unit*" sendo considerada a primeira GPU do mundo (NVIDIA Corporation, 2013). As GPUs passaram a ter um papel importante na geração de gráficos 2D e 3D aliviando a carga das CPUs. Então através de programação de baixo nível, cientistas de computação e outros procuraram aproveitar a capacidade de processamento existente das GPUs para outros fins já que boa parte do tempo a maioria delas ficava ociosa e alguns atingindo sem precedentes performance de mais de 100 (cem) vezes em comparação com CPUs (NVIDIA Corporation, 2014b).

Até então, utilizar as instruções programáveis das GPUs não era tarefa fácil, a melhor forma era utilizar APIs (Interfaces de Programação de Aplicativos) para programação, por exemplo a OpenGL. A NVIDIA percebendo a necessidade e por ser uma das principais fabricantes de GPUs acabou ajudando no processo criando bibliotecas C, C++ e Fortran que facilitaram o uso da GPGPU. Desta forma, foi inventada a plataforma CUDA - *Compute Unified Device Architecture* (NVIDIA Corporation, 2014a) (A tecnologia CUDA será abordada mais adiante, no capítulo 2.5.3).

Com a plataforma CUDA, esta linha de trabalho passou a ser mais acessível a todos os programadores da linguagem C++. As atualizações da plataforma também são constantes, desde fevereiro de 2014 está disponível a versão 6 com material acessível através do site do fabricante. Já é bem mais fácil nos dias de hoje fazer download da ferramenta através do site www.nvidia.com, ter uma placa NVIDIA e explorar a computação paralela com GPU, os programadores não precisam mais utilizar programação de tão baixo nível como antes.

Logo após o lançamento da plataforma CUDA, que é específica para GPUs da NVIDIA, foi criada também uma API chamada OpenCL, uma forma mais genérica de programação para GPGPU e independente do fabricante da placa de vídeo.

Com o decorrer do tempo foram criados modelos de GPUs mais específicos e adequados para GPGPU, como as placas de vídeo Tesla para supercomputação e os modelos TEGRA que são mais apropriadas para dispositivos móveis como *tablets* e

celulares. Pela primeira vez na história usuários de dispositivos móveis podem se beneficiar do processamento paralelo na palma da mão aproveitando a GPGPU. O ganho é visível para aqueles que usam algum software cuja utilização da GPU é evidente no desempenho.

2.3 A ARQUITETURA DA GPU PARA COMPUTAÇÃO PARALELA

Uma das perguntas que todos sempre acabam fazendo é sobre o fato de que é possível usar o processamento da GPU mas por que não pode ser usada igual a CPU? O que difere um núcleo CUDA do núcleo de uma CPU? O poder de processamento está lá na GPU e é notado que acaba ficando muitas vezes ocioso sendo que a CPU fica várias vezes no limite de 100% de processamento.

Os *cores* (núcleos) da GPU possuem uma arquitetura diferente de uma CPU tradicional, a CPU executa diversos processos que são alocados e gerenciados pelo Sistema Operacional e enquanto a CPU executa um processo e depois outro, a GPU possui uma característica de processamento em blocos, sua tarefa é uma forma de alocação de dados e processamento em conjunto, todos eles ao mesmo tempo porque estes blocos são especializados em rasterização de pixels.

No capítulo 4.2 será abordado com mais detalhe este assunto, como será visto talvez uma das principais diferenças esteja no que é chamado de *warp*, um “momento” em que são executadas em paralelo 32 *threads*, cópias de uma mesma instrução.

2.4 NEM TODO PROGRAMA PODE SER ACELERADO POR GPU

A capacidade de acelerar o processamento usando GPGPU está relacionada com a arquitetura da GPU. A forma de pensar uma solução usando a CPU ou criação de *threads* para uma CPU é bem diferente da alocação de recursos de uma GPU. A aceleração depende de como é programada a solução para o problema que se apresenta. Muitos algoritmos de processamento gráfico são bons candidatos para usar GPGPU porque para aplicações que processam imagens são comuns algoritmos que exigem que o mesmo procedimento de cálculo seja aplicado a todos os pixels da

imagem sem que o processamento de um pixel dependa do resultado do processamento de outros. Problemas que precisam da execução de uma função em um elemento por vez (um elemento de um vetor por exemplo) e que não pode ser aplicada a todos os elementos ao mesmo tempo, não costuma ter grandes ganhos através da GPGPU. Cada caso é uma situação a ser avaliada, a GPGPU é altamente eficiente para solução de uma série de problemas mas não de outros.

Os capítulos 4 e 5 focam casos de uso com a GPGPU, o objetivo é trazer um entendimento para identificação de quais tipos de problemas são melhores para GPGPU.

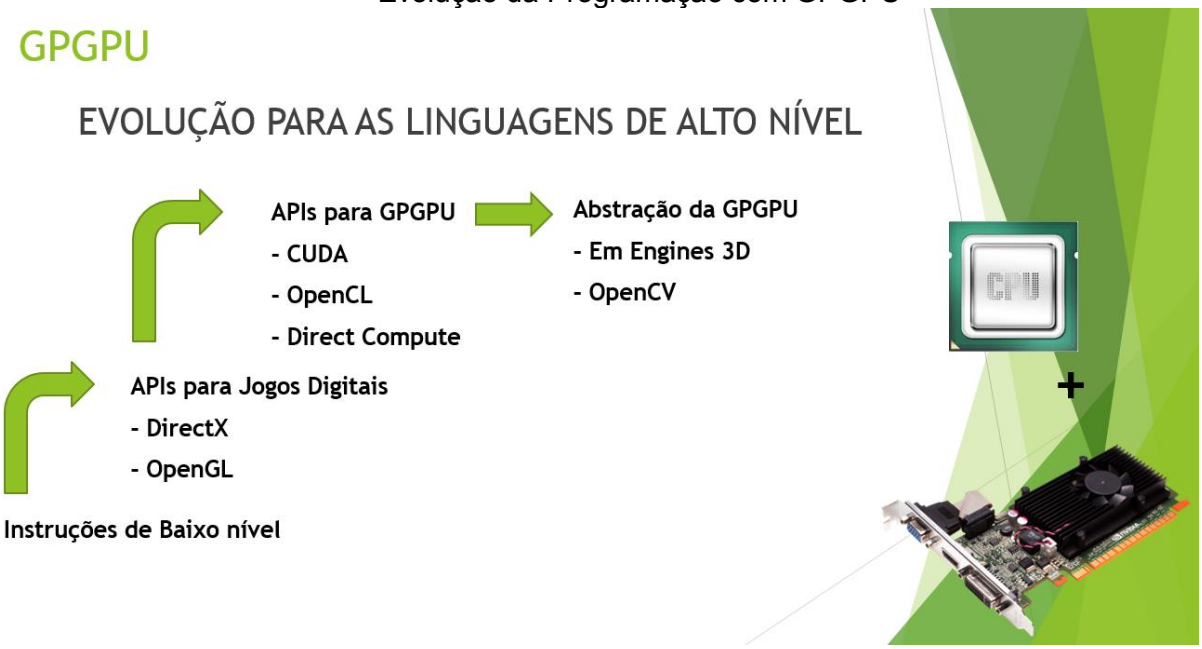
2.5 API – INTERFACE DE PROGRAMAÇÃO DE APLICATIVOS

Formas geométricas, iluminação, cores, filtros e texturas são as matérias-primas de todo e qualquer vídeo game. São eles os componentes que, quando combinados com movimentos, efeitos e enredos, produzem os mais diversos jogos eletrônicos.

Para que a experiência de um jogo possa ser transmitida ao jogador, é necessário que o desenvolvedor primeiro programe o “aparecimento” de todos esses efeitos gráficos no monitor, bem como as sequências de interações e reações do programa.

Tal tarefa era considerada uma empreitada colossal nos primórdios da computação, mas foi ficando cada vez mais facilitada com o surgimento das APIs - Interfaces de Programação de Aplicativos. Algumas destas APIs relacionadas com Jogos Digitais são a DirectX® (MICROSOFT, 2014) e a OpenGL (Khronos Group, 2014a). E recentemente (2008) surgiu uma outra API chamada OpenCL (Khronos Group, 2014b) que tem mais haver com este nosso trabalho em estudo, pois é voltada para a programação genérica através de GPUs. A figura 2 exemplifica a evolução da programação da GPGPU, no início eram usadas as instruções de baixo nível, depois surgiram as APIs e as plataformas específicas para GPGPU.

Figura 2
Evolução da Programação com GPGPU



Fonte: Próprio autor

2.5.1 DirectX®

A tecnologia DirectX® surgiu de forma a facilitar o trabalho do programador, usar uma interface de instruções sem a necessidade de saber em qual modelo do dispositivo de vídeo as mesmas são executadas. É um conjunto de APIs que foi desenvolvido pela empresa Microsoft® para funcionar em seus Sistemas Operacionais Windows. A tecnologia DirectX® é composta por várias APIs voltadas para determinadas tarefas (WIKIPEDIA, 2014), por exemplo:

- Direct Draw: para desenhos de gráficos 2D.

- Direct 3D: para desenhos de gráficos 3D.

- Direct Input: para distribuição de dispositivos de controle – teclados, mouses, joysticks, ou outros controladores de jogos.

- Direct Play: para comunicação de rede local de computadores ou internet.

- Direct Sound: para a reprodução e gravação de sons no formato *wave*.

- Direct Sound 3D: para a reprodução de sons 3D.

- Direct Compute: para GPGPU (inclusão no DirectX® 11).

É possível perceber que o conjunto de APIs DirectX® não é exclusivo para Placas de Vídeo, é destinado para vários tipos de dispositivos como Placas de Som, Placas de Rede e outros.

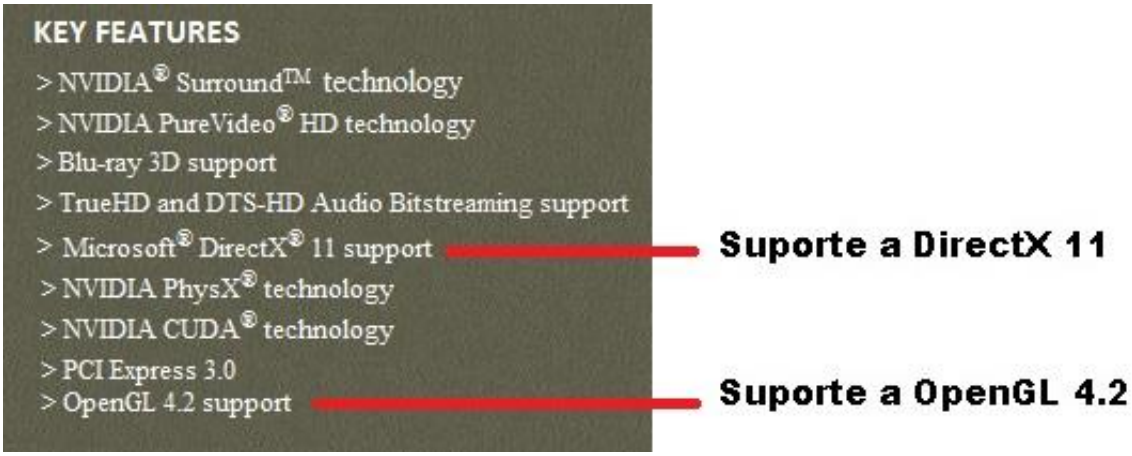
Além da vantagem do programador não precisar saber exatamente qual hardware está instalado, é uma API que faz acesso direto ao hardware, não necessitando passar por outras camadas de software (drivers por exemplo). Por causa do acesso direto ao hardware, sempre foi excelente para performance dos jogos digitais.

Em maio de 2014, as versões disponíveis da DirectX® eram: Para Windows XP, a DirectX® 9.0c; para Windows Vista e Windows Seven, a DirectcX® 11.0 e para Windows 8, a DirectX® 11.1 (nota: o Windows XP é um Sistema Operacional considerado *deprecated* (obsoleto), já que a Microsoft encerrou o suporte a ele em abril de 2014).

Os fabricantes costumam trazer informações nas especificações acerca do suporte a DirectX®, menções como por exemplo “*Microsoft® DirectX® 11 Support*” e assim é possível identificar, por exemplo, se há compatibilidade para um determinado jogo que exija DirectX®.

Imagem 3

Na embalagem da GeForce GT 640, compatibilidade com DirectX® 11.



Fonte: Próprio Autor.

Para o desenvolvimento em outros Sistemas Operacionais que não são Microsoft Windows e portanto sem suporte a DirectX®, como Linux ou MAC OS, uma alternativa é a utilização de uma outra API chamada OpenGL.

2.5.2 OpenGL

O aparecimento da API OpenGL também é muito similar ao da API DirectX. Na década de 80, cada fabricante de hardware tinha seu próprio conjunto de instruções para desenho de gráficos 2D e 3D. Construir aplicações com essas tecnologias que suportassem vários hardwares era um verdadeiro desafio. O esforço era constantemente duplicado e havia pouco espaço para companhias menores e sem capital para tamanho investimento.

É importante saber que com o resultado de vários trabalhos, muitos vindos da SGI - Silicon Graphics, a API OpenGL foi criada em 1992 (Khronos Group, 2014). Desde 1992, o padrão é mantido pelo ARB - Architecture Review Board, um conselho formado por empresas como a Silicon Graphics, 3DLab, ATI, Dell, Evans&Sutherland, HP, IBM, Intel, Matrox, NVIDIA, Sun e outras. O papel desse conselho é manter a

especificação e indicar quais recursos serão adicionados a cada versão. Em maio de 2014, a versão do OpenGL era a 4.4.

É também interessante saber que praticamente todas as modificações recentes do padrão começaram como extensões, a grande maioria devido ao mercado de Jogos Digitais. Extensões aqui são funções que o fabricante adiciona na Placa de Vídeo mas que não faz parte das funções padrões comuns a toda placa de vídeo, são funções portanto só presentes naquela placa daquele fabricante e muitas delas mais tarde acabam sendo inseridas nas funções padrões.

Basicamente a OpenGL é um conjunto de algumas centenas de funções, que fornecem acesso a praticamente todos os recursos do dispositivo de vídeo.

Os seguintes sites sempre trazem informações oficiais atualizadas sobre OpenGL: http:\\www.sgi.com (da Silicon Graphics) e http:\\www.opengl.org.

Para saber qual versão de OpenGL uma placa de vídeo suporta, verificar nas especificações do produto. Geralmente vem algo assim "*OpenGL 4.2 support*" similar ao caso da DirectX®.

2.5.3 A TECNOLOGIA CUDA

A NVIDIA é uma empresa líder na fabricação de placas de vídeo e precursora no desenvolvimento da computação genérica através de GPUs. Foi pioneira na distribuição pública de um conjunto de ferramentas para que qualquer programador do planeta tivesse acesso mais fácil no desenvolvimento da GPGPU com os seus produtos, a estas ferramentas deu o nome de plataforma CUDA - *Compute Unified Device Architecture* (NVIDIA Corporation, 2014a). A plataforma CUDA é constituída de diversas bibliotecas para serem utilizadas com a linguagem C/C++ ou Fortran, com funções de mais alto nível para o programador usar em GPGPU. A plataforma pode ser pega através da área de downloads do fabricante. A plataforma CUDA é gratuita. Há versões do "*CUDA Toolkit*" para Windows, Linux e MAC OS.

Os núcleos das placas de vídeo da NVIDIA ganharam o apelido de “CUDA cores” devido ao suporte à plataforma CUDA. A tabela 1 mostra alguns modelos e a respectiva quantidade de “CUDA cores”

Vários desenvolvedores, aproveitando as instruções CUDA, também desenvolveram outras bibliotecas para acesso de mais alto nível a diversos programadores, um bom exemplo é a biblioteca para OpenCV específica para plataforma CUDA. Através do site do fabricante é possível ver outras disponíveis além dessa (NVIDIA Corporation, 2014c).

Tabela 1

Modelos de GPUs da família GTX e seus “CUDA” cores

GPU	“CUDA cores”
GTX 750	512
GTX 750 TI	640
GTX 760	1152
GTX 770	1536
GTX 780	2304
GTX 780 TI	2880

Fonte: Próprio autor. Dados coletados de especificações dos produtos da fabricante NVIDIA.

É Importante destacar que muitos softwares de mercado já usam a plataforma CUDA (e também a API OpenCL) como softwares para CAD, Engines de Jogos e muitos outros. Estas aplicações aproveitam esta tecnologia se o usuário tiver em seu computador uma placa de vídeo compatível.

2.5.4 OpenCL

OpenCL (*Open Computing Language*) (Khronos Group, 2014b) é uma padronização surgida em 2008, muito similar a ideia do OpenGL, mas a OpenCL é voltada para a programação paralela, e não só para placas de vídeo mas também para outros dispositivos.

A programação paralela com OpenCL em Placas de Vídeo é muito parecida com a forma de programação da plataforma CUDA.

Igualmente a OpenGL é possível encontrar produtos no mercado dizendo que possuem suporte a OpenCL 1.2 ou a OpenCL 2.0 (ver nas especificações do produto). Cada nova versão vai agregando novas funções ou modificando outras. Eventualmente é possível encontrar um software cujo requisito seja OpenCL 2.0 para um determinado dispositivo e então seu hardware deverá pelo menos ter suporte na versão 2.0. Logo há a necessidade de um certo cuidado para a aquisição de um produto compatível com o software disponível.

Verifica-se também que as placas de vídeo da NVIDIA não possuem por enquanto total suporte a OpenCL. O melhor é a plataforma CUDA de programação neste caso. É possível perceber então que existe um problema em relação à tecnologia que o desenvolvedor vai escolher, para atender mais de um fabricante ao mesmo tempo pode ser necessário utilizar mais de uma tecnologia de programação ou então optar por produto de somente um fabricante.

2.5.5 OpenCV

É uma conceituada coleção de códigos para Visão Computacional (OpenCV – *Open Source Computer Vision*). Existe um conjunto de códigos que foi “turbinado” para aproveitar a GPGPU e desde 2010 há um novo módulo escrito para fins de aceleração através de GPU que tem crescido desde então. Possui certa compatibilidade com OpenCL 2.0 e com a plataforma CUDA.

> “Com a adição da aceleração CUDA ao OpenCV, desenvolvedores podem executar algoritmos mais sofisticados e com mais acuidade em tempo real e com imagens de alta resolução enquanto é consumida menos energia” (OpenCV,2014, tradução nossa).

Na utilização do OpenCV é plausível aproveitar computadores com Placas de Vídeo, estas aplicações podem ser muito mais eficientes com o aproveitamento de uma GPU. São alguns exemplos de aplicações com OpenCV: Visão Computacional em Tempo Real, Processamento de Imagens, Reconhecimento Facial e Realidade Aumentada.

2.6 EXEMPLOS DE SOFTWARES QUE USAM GPGPU

Abaixo está uma relação de alguns programas conhecidos que fazem uso da GPGPU (NVIDIA Corporation, 2014e). É importante lembrar que a adoção de uma placa de vídeo compatível para execução destes programas pode trazer um benefício realmente grande aos profissionais e/ou empresas que os utilizam, alguns casos podem representar diferenças de apenas minutos de processamento, mas outros podem representar horas ou dias e respectivamente podem impactar em valores financeiros envolvidos e consequentemente na concorrência de mercado. A escolha do hardware adequado então faz diferença.

- MATLAB

- AutoCAD

- Autodesk 3ds Max

- Inventor

- Maya

- Adobe After Effects

- GEOS 5

- Einstein@Home

3 GPU, GPGPU E JOGOS DIGITAIS

Jogos Digitais sempre estiveram relacionados com placas de processamento gráfico, pois sempre dependeram delas para melhor apresentação das imagens, dos ambientes virtuais, personagens e objetos ao jogador. Agora os jogos também começam a aproveitar recursos através da GPGPU e alcançar interações homem-máquina antes impossibilitadas por falta de processamento de máquina. Este capítulo procura mostrar como ao longo do tempo vem acontecendo esta proximidade intrínseca dos Jogos Digitais com as GPUs e como a GPGPU está se inserindo no meio. É possível ter uma noção aqui de como vem evoluindo as linguagens de programação e as instruções se tornando de mais alto nível para o desenvolvedor de jogos.

3.1 A CRIAÇÃO DE JOGOS DIGITAIS E A GPU

As Unidades de Processamento Gráfico sempre foram peças importantes para jogos de computador, quanto mais potentes melhores gráficos, então sempre houve uma estreita relação entre jogos e placas de vídeo e assim é até hoje.

Assim como a algum tempo atrás a GPGPU se utilizava apenas de programação de baixo nível, os programadores de jogos também precisavam instruções de baixo nível para usar os recursos das placas de vídeo. Com o passar do tempo o acesso a esses recursos foi ficando mais fácil com o surgimento de bibliotecas de mais alto nível, em um primeiro momento e depois engines para jogos como Blitz 3D (BLITZ Research, 2014), Unity 3D (Unity Technologies, 2014), Unreal Engine (EPIC GAMES, 2014) e outros, nestas engines muitas instruções já ficam abstraídas, como por exemplo as da DirectX®.

É interessante entender como se formaram alguns padrões de mercado, nas primeiras placas não havia um padrão comum entre fabricantes para acesso às instruções disponíveis nos dispositivos, por isso a Microsoft desenvolveu o DirectX, um conjunto de APIs que o desenvolvedor de jogos pode utilizar não se importando como é a instrução do fabricante para a placa de vídeo, o desenvolvedor apenas chama sua função de mais alto nível do DirectX que abstrai código de mais baixo nível.

O mercado também criou entre os fabricantes uma outra API, a OpenGL, instruções padrões disponibilizadas em comum nas placas de vídeo, as placas passaram então a trazer informações sobre qual versão de OpenGL suportam, assim uma placa suporta o padrão OpenGL 2.0 enquanto outra mais nova já pode oferecer um suporte a versão de instruções OpenGL 3.0 por exemplo. Bibliotecas OpenGL em C++ já foram e ainda são uma boa opção aos desenvolvedores. Como o DirectX é algo exclusivo do Windows, nas Plataformas Linux os jogos costumam utilizar o padrão OpenGL para acesso aos recursos das Placas de Vídeo, pois OpenGL atende a uma variedade de plataformas (Khronos Group, 2014a).

Atualmente há implementações de classes em C++ e em outras linguagens que abstraem instruções como da OpenGL, DirectX e outras APIs consideradas de programação de mais baixo nível. Isto leva o desenvolvedor de games a não precisar se preocupar mais com estas APIs. *Engines* modernas 2D e 3D também já possuem as APIs embutidas e transparentes ao desenvolvedor, este apenas utiliza a instrução de mais alto nível oferecida pela *engine.*

Observa-se que a tendência é que as engines de jogos e outras ferramentas de desenvolvimento também passem a abstrair instruções para GPGPU, as engines Unity3D e Unreal Engine já embutem de forma transparente ao programador recursos de GPGPU. A intenção é que instruções como laços "*for ... next*" depois de compiladas, passem a utilizar a GPU caso a mesma esteja disponível. Instruções de Processamento de Reconhecimento Facial são um bom exemplo.

3.2 APLICABILIDADE DA GPGPU EM JOGOS DIGITAIS

A Computação com GPUs traz enormes benefícios para uma diversidade de áreas envolvendo computação (NVIDIA Corporation, 2014e) e o Profissional de Jogos Digitais atua em diversas destas áreas. A seguir, algumas das utilizações na área.

3.2.1 Simulações

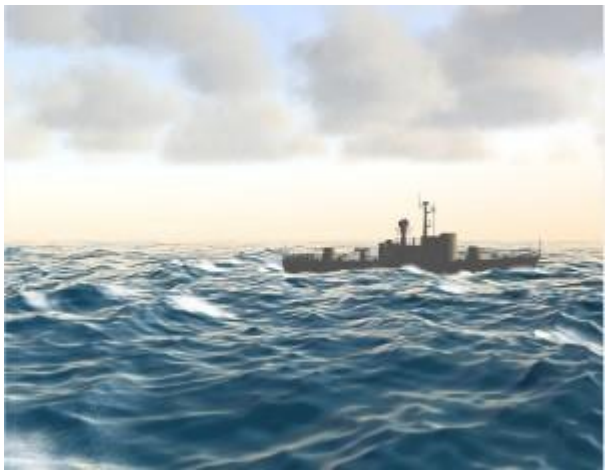
Simulações costumam envolver grande esforço computacional, quanto mais detalhada é a simulação, mais processamento de máquina será exigido, movimentos de água e efeitos de vento são um bom exemplo. Há jogos em que se buscam ambientes mais realistas para maior imersão do jogador. O Profissional de Jogos

precisa ter a percepção de qual ferramenta utilizar para alcançar o seu objetivo, precisará avaliar quais serão as opções mais adequadas para um produto que seu cliente deseja, alternativas que utilizam GPGPU podem ser a solução.

O Profissional de Jogos Digitais pode criar simulações não só para fins de entretenimento, pode também montar situações para aprendizagem, pesquisa científica, para previsões como em outras aplicações sérias que fazem previsão do tempo, em todas estas linhas a GPGPU pode inclusive tornar viável o que antes não era.

Simulações estão presentes dentro de Jogos Digitais como também na criação dos vídeos que fazem parte das apresentações ao jogador (*cinematics*) além de outras apresentações para comerciais diversos. A imagem 4, mostra uma simulação de movimento da água realizado com o SDK Triton (Triton™ é um produto da Sundog Software disponível para utilização com Unity 3D):

Imagem 4
Simulação com movimentos da água realizada com o SDK Triton utilizando GPGPU



Fonte: (Sundog Software, 2013).

3.2.2 Ambientes Virtuais

Podem ser ambientes virtuais para entretenimento, um jogo por exemplo como o famoso *World of Warcraft*, ou para fins de demonstração de um produto como a visita virtual a um apartamento a venda, ou ainda para fins de treinamento...

A RV – Realidade Virtual, está muito presente em jogos e em vários outros softwares.

3.2.3 Reconhecimento de Imagens

Várias aplicações modernas envolvem RA, Realidade Aumentada, e precisam a todo momento reconhecer marcações em imagens. Novas formas de interação homem-máquina como o Kinect da Microsoft, onde o software percebe através de uma câmera movimentos do jogador e realizada a interatividade e outras aplicações envolvendo *Perceptual-Computing*[2] são bons exemplos.

Para dispositivos móveis com reconhecimento de imagens, podendo ser celulares ou outros dispositivos novos como óculos acoplados a câmeras, a GPGPU pode tornar viável o que antes não era por causa da necessidade da resposta em tempo real que o dispositivo precisa dar ao usuário.

3.2.4 Desenvolvimento de *Engines*

Engines para jogos como Unity 3D e UDK, já dispõe em suas últimas versões de forma transparente ao desenvolvedor os recursos de GPGPU. Um mesmo jogo compilado com o Unity 3D de uma versão anterior pode apresentar diferença perceptível ao jogador em máquinas com placas de vídeo compatíveis para GPGPU quando compilados nas versões atuais que possuem a tecnologia embutida.

3.2.5 Supercomputação para Simulações Sérias

Bons exemplos de sucesso são: Simulações Sísmicas da PETROBRAS, Simulações de TSUNAMES e Previsão do Tempo.

Geralmente estas simulações demandam muito esforço computacional e o custo do hardware utilizando muitas GPUs é muito menor para montar supercomputadores comparando com CPUs.

---

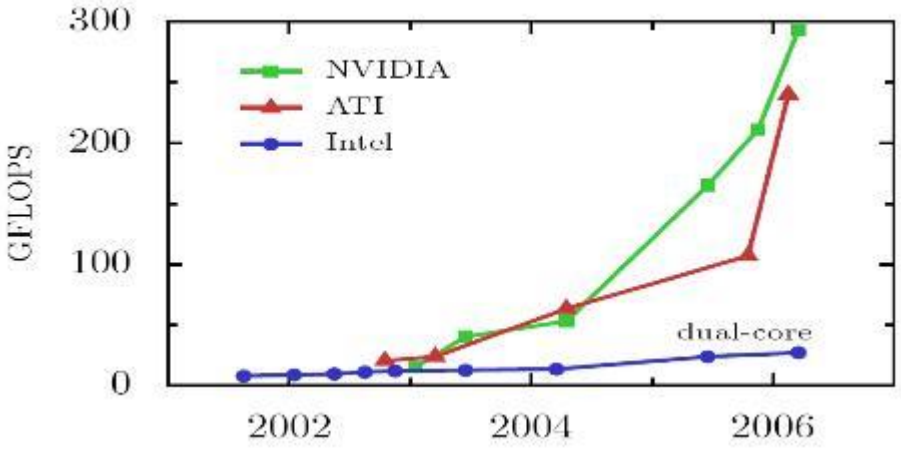
[2] *Perceptual-computing*: técnica computacional que envolve a UI – *User Interface* (Interface com o Usuário) através da percepção dos movimentos naturais do corpo.

4 ANALISAR O PROBLEMA E ESCOLHER O HARDWARE ADEQUADO

Apenas considerar alguns gráficos de capacidade de processamento entre CPU e GPU (ver gráfico 1) que circulam na internet, pode levar o usuário ou mesmo um profissional a erro. Estes gráficos geralmente apontam para o fato da capacidade de processamento das GPUs superarem as CPUs a um custo menor. A questão é que isto não significa que o software que o usuário usa será executado duas ou três vezes mais rápido. Na verdade depende do problema em questão para ser resolvido computacionalmente. Há certo grupo de problemas que a GPGPU pode reduzir consideravelmente o tempo para se chegar a um resultado, mas isso não se aplica a todo e qualquer problema. Outra questão é que a GPGPU não depende somente da GPU, a CPU também faz parte do trabalho computacional.

Gráfico 1

Evolução da Capacidade de GFLOP/s, CPU Intel X GPUs



Fonte: Fantalgo, 2014.

4.1 PROBLEMAS QUE “PEDEM” GPGPU

Fato: De um lado CPUs são rápidas (3GHz), e executam as instruções sequencialmente. Processadores modernos como o i7 são rápidos e podem executar até 8 threads ao mesmo tempo porque possuem 4 núcleos *multithreading*[3]. De outro

[3] *Multithreading*, tecnologia embutida em modelos de processadores da Intel a partir do Pentium 4, onde um núcleo do processador pode executar simultaneamente 2 threads.

lado os núcleos das GPUs são de mais baixa velocidade (400MHz por exemplo) mas executam vários blocos de dados ao mesmo tempo, uma GPU pode ter mais de 1024 núcleos ao mesmo custo de uma CPU moderna de 4 núcleos.

O segredo da GPGPU está no processamento executado em paralelo, ao mesmo tempo com todos os dados de um bloco, ou de vários blocos ao mesmo tempo. Levando isso em consideração, problemas que envolvem matrizes de dados e existindo funções matemáticas que podem ser executadas sobre todos os dados simultaneamente, são problemas adequados para GPGPU. Nesta categoria de problemas há, por exemplo, o processamento de imagens. Considerando, por exemplo, que em uma imagem colorida seja aplicado um algoritmo para transformá-la em tons de cinza, um cálculo precisa ser executado igualmente em todos os pixels da imagem, aí então está uma situação favorável para uso da GPGPU.

Agora considerando um outro problema, um jogo de xadrez em que uma peça se move no tabuleiro e um cálculo então que é feito para saber se a peça está ou não em uma boa posição, que é necessário avaliar também outras possibilidades de posições que a peça possa se deslocar. Para este problema não há como fazer o cálculo ao mesmo tempo das várias posições, a situação do tabuleiro muda e as variáveis em volta da peça precisam ser recalculadas. Este então não é um bom problema para GPGPU a não ser que alguém invente alguma outra técnica de avalição posicional para a peça de xadrez. Soluções em árvores se tornam mais eficientes usando *threads* com várias CPUs (ver capítulo 5.2).

Nota-se que problemas como o de transformar a imagem em tons de cinza, aproveita-se muito melhor o bloco na GPU, é possível aproveitar melhor as posições do bloco e a função matemática ser aplicada na massa de dados e não em apenas um dado. E também quanto menos trocas de dados entre a memória RAM e a GPU melhor. Se uma massa de dados já está na GPU e não precisar ser deslocada para a memória RAM, sendo executadas várias outras vezes funções matemáticas sobre o que já está na GPU, a GPGPU torna-se mais eficiente. É possível perceber estes fatos após estudos de programação com a plataforma CUDA (NVIDIA Corporation, 2014f).

Assim, é possível perceber que várias categorias de problemas como reconhecimento de imagem, simulações de terremotos, previsões climáticas, problemas cujas soluções podem ser pensadas com matrizes, seja representando superfícies ou volumes são mais adequadas ao uso da GPGPU do que soluções cujos

cálculos precisam ser lineares dado a dado. É necessário para eficiência da GPGPU que o algoritmo usado na solução possa ser executado em paralelo nos dados.

O profissional ao planejar um computador adequado para o seu propósito, deve então analisar primeiro para qual finalidade será destinado. Podemos ter por exemplo, um supercomputador A com várias CPUs a um custo de R$ 500.000,00 e do outro lado um supercomputador B com várias GPUs a um custo de R$ 100.000,00, sendo que este pode ter eficiência 20 vezes superior ao supercomputador A dependendo do programa que será executado. Não é sempre certo dizer que A é melhor que B ou que B é melhor que A, precisa ser considerado o custo-benefício com o trabalho que será executado (ver como exemplo o capítulo 4.3, caso da PETROBRAS).

O problema em questão então precisa ser analisado, pode não ser considerado viável para ser criada uma solução por GPGPU para o mesmo.

Um outro fato a ser pensado é que a programação via GPGPU não segue uma linha convencional de programação em *threads* que rodam em CPUs, o desenvolvimento de software geralmente é mais demorado e gasta mais recursos de horas de programação e tempo para o produto final. Uma solução que exista de forma linear para um problema precisa ser repensada e a solução precisa ser reescrita para que os blocos na GPU sejam aproveitados adequadamente, então aí está um ponto a pensar sobre custo-benefício, o produto final realmente precisa da solução via GPGPU? Geralmente se é um produto das categorias acima em que a solução via GPGPU é muito mais eficiente e se é um produto que fica funcionando 24 horas por dia executando massa de dados, provavelmente será crucial o uso da GPGPU.

4.2 A ARQUITETURA DE UMA GPU

Como visto no item anterior, é necessária uma avaliação do problema para alocar uma solução em bloco e assim aproveitar melhor a GPGPU, isso leva à necessidade de conhecermos melhor o que de fato acontece na execução das *threads* da GPU.

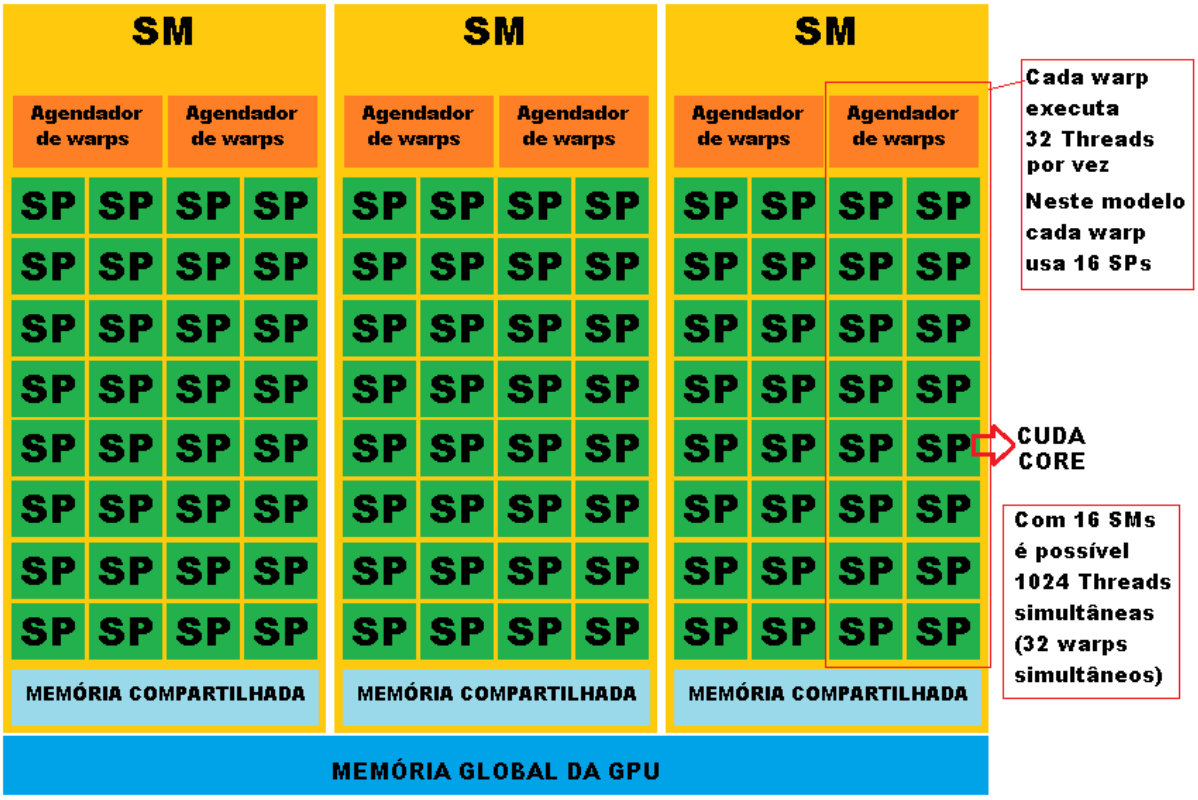
Para exemplo, vamos considerar uma das placas de vídeo, a GTX 580 da NVIDIA com 512 CUDA cores, arquitetura Fermi (NVIDIA Corporation, 2014d). Ela é composta de 16 SM (*Streaming Multiprocessor*) e cada SM possui 32 SP (*Streaming*

Processor), totalizando 16 x 32 = 512 *Streaming Processors* (que corresponde a 512 CUDA cores).

Criamos a figura explicativa abaixo com o intuito de entendimento do significado de um *Streaming Multiprocessor* e seus vários *Streaming Processors*. No caso da GTX 580 são 16 SMs como o da figura totalizando 512 SPs (16x32).

Imagem 5

Representação simplificada de SMs (*Streaming Multiprocessors*) da tecnologia Fermi.



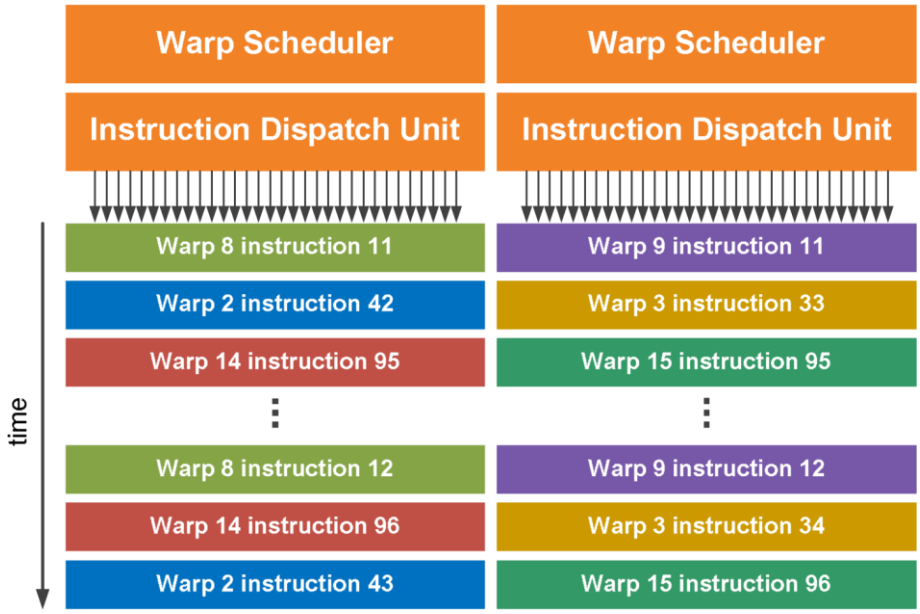
Fonte: Próprio autor baseado na arquitetura Fermi (NVIDIA Corporation, 2014d)

Antes de executar um programa na GPU é preciso definir a lógica de execução de dados no que é chamado na GPGPU CUDA de “blocos”, podemos definir vários blocos. A GPU aloca estes blocos nos *Streaming Multiprocessors* e depois são executados em *warps*. Cada warp processa 32 threads. Na imagem acima pode-se observar que cada SM consegue executar 2 warps de cada vez e como a GPU GTX 580 possui 16 SMs, isso significa a execução de até 32 *warps* simultâneos, ou 1024 *threads* simultâneas. É importante enfatizar que um *warp* possui a capacidade de executar apenas um único código (como são 16 SPs sendo usados naquele *warp*, significa16 cópias do mesmo código para execução física em paralelo), somente após

a execução de um *warp* é que é possível ter outra instrução para o próximo *warp*, então por aí pode-se deduzir que o aproveitamento do *warp* é fazer com que sejam executadas 32 cópias da mesma função ao mesmo tempo, se for executada apenas 2 cópias da função, significa que o *warp* ocorreu de qualquer forma mas só foram aproveitadas 2 *threads* das 32 que eram possíveis naquele momento (percebemos que é neste ponto que algoritmos como do OpenCV ganham vantagem com GPGPU, porque pela massa de dados é possível tentar executar o máximo possível de cópias de uma mesma função ao mesmo tempo, sem desperdício do *warp*).

Imagem 6

Execução de Warps em um SM Fermi



Fonte: Fermi - *Dual Warp Scheduler* (NVIDIA Corporation, 2014d, p. 10)

Pela quantidade de SM e SP em uma placa de vídeo é possível perceber que é necessário adequar o código de forma dinâmica a fim de otimizar o processamento, conforme o conjunto de instruções, pode ser usada a lógica de execução de vários blocos em paralelo ou concentrar mais threads para um mesmo bloco, a GTX 280 por exemplo tem 30 SM cada um com 8 SP, significando que e a quantidade de threads e blocos em paralelo mudam conforme cada placa de vídeo.

As funções nesta programação são chamadas de *kernels* (NVIDIA Corporation, 2014f), tecnicamente então dizemos que em um *warp* ocorre a execução de um *kernel*, este *kernel* no *warp* pode ser executado até 32 vezes simultaneamente. Havendo

um *kernel* diferente para ser executado, ele terá que esperar um próximo *warp* ou ser executado em um outro warp que pode estar ocorrendo dentro do mesmo SM ou em outro SM paralelamente. A montagem do código para que *kernels* diferentes (ou mesmo que sejam iguais) sejam executados em blocos diferentes para que sejam executados em paralelo, se chama de execução paralela em blocos, enfim é possível combinar a execução paralela de threads de um bloco com outros blocos ao mesmo tempo. Algoritmos que sejam possíveis de implementar as instruções para aproveitarem os *warps* e os blocos é que tornam a GPGPU atraente. Logo é possível perceber que realmente todo tipo de código não é possível implementar com eficiência em uma GPU, há aqueles mais e outros menos adequados.

Na prática o fluxo básico de programação da GPGPU é primeiro copiar os dados da memória RAM do computador para a memória da GPU, depois copiar para a GPU a função que será executada e após a execução, coletar os resultados da memória da GPU para a memória do computador (TARIQ, 2011) (ver imagens do Apêndice 1).

Há outras características em um SM que não estão sendo detalhadas aqui, estamos considerando apenas o que é mais importante para o entendimento de como alocar com eficiência a execução dos *kernels* em uma GPU (aproveitar as 32 *threads* que ocorrem em 1 *warp*). Outra coisa que deve ser considerada, é o tamanho de memória compartilhada que todas as SPs de um mesmo SM podem acessar ao mesmo tempo, memórias que estão em um SM não são acessíveis por outro SM sem haver uma perda de performance.

4.3 SUPERCOMPUTADORES COM GPUS

Como citado, para alguns tipos de problemas é mais adequado o uso de GPGPU e por isso existem atualmente vários supercomputadores no mundo baseados em GPUs para fins diversos.

É possível montar pequenos “supercomputadores domésticos” com apenas uma placa de vídeo, ou adicionando várias em um mesmo computador. Se houver várias interfaces PCI Express na placa mãe de um computador, é possível equipá-lo com várias GPUs ao mesmo tempo.

Para aplicações mais robustas, entram em cena os conjuntos de vários computadores com várias GPUs trabalhando juntos. Abaixo está um caso bem brasileiro, o supercomputador da PETROBRAS que é usado para simulações sísmicas. Ocupou em 2012 a posição 68 da lista *TOP* 500.

Em princípio a simulação sísmica envolve cálculos sobre a superfície terrestre e volumes o que nos leva a pensar que pode ser possível o aproveitamento da GPGPU já que vários cálculos podem ser aplicados ao mesmo tempo sobre uma massa de dados. É isso que ocorreu para ser desenvolvido o Grifo4.

> “O Grifo04 foi projetado pela equipe de tecnologia da informação (TI) da PETROBRAS, em parceria com o grupo de exploração e produção (E&P) e custou para a estatal R$ 15 milhões. "O que motivou a busca por outra tecnologia foi perceber que o espaço estava acabando e o consumo de energia já estava alto", afirma Carlos Henrique de Albrecht, analista de sistemas sênior da Petrobras. Ele diz que o Grifo04 consome 90% menos energia que um supercomputador vendido no mercado atualmente. ” (SEESP, 2014)

> “A tecnologia solicitada pela PETROBRAS e que mudou o paradigma da computação é a de processadores gráficos (GPUs), especializados em manipular imagens, vídeos e gráficos. Eles são capazes de realizar vários cálculos matemáticos de forma simultânea. Além disso, as informações são processadas sem passar pelo processador central (CPU). Com isso, os equipamentos são mais velozes que computadores que só usam a CPU.” (SEESP, 2014)

Imagem 7
Supercomputador Grifo4

Fonte: PETROBRAS – Fatos e Dados (PETROBRAS, 2014)

O estudo de caso do Grifo4 é um bom exemplo ligando o problema certo ao aproveitamento eficiente de GPGPU, na economia de energia. E na questão de custo de equipamento a aquisição na época ficou em R$ 15 milhões contra o estimado de mercado que era de R$ 180 milhões.

5 GPGPU - EXPERIMENTOS PRÁTICOS

Para este trabalho foram realizados 2 experimentos práticos pelo próprio autor em computadores diferentes. Em cada um dos experimentos foram consideradas tarefas sendo executadas simultaneamente, sempre cada uma utilizando 50% dos recursos de CPU mas uma delas com algoritmos de GPGPU. Através do gerenciador de tarefas do Windows foi monitorada a utilização dos recursos da CPU para certeza de ambas as tarefas competirem igualmente. Durante o procedimento, estas tarefas apresentaram oscilação de até 1% na utilização da CPU, isto é, cada tarefa apresentou utilização entre 49% e 50% da CPU sendo considerado desprezível no procedimento. As aplicações utilizadas trazem informações de GFLOPs processados em cada tarefa e o tempo de processamento, desta forma foi feita uma comparação da velocidade de processamento medida em GFLOP/s entre as duas estratégias – GPGPU e somente CPU. As aplicações utilizadas, SETI@home e Einstein@home já são conhecidas por apresentarem melhor desempenho com o uso da GPGPU. Os problemas envolvidos nestas aplicações são da categoria considerada favorável ao uso de GPGPU por causa do paralelismo possível com seus dados.

Para comparar problemas considerados não favoráveis ao uso de GPGPU (problemas cuja massa de dados não é favorável para execução em paralelismo), foi considerado o problema N-Rainhas e a comparação entre CPU e GPU em milissegundos para o programa encontrar as soluções (PAMPLONA, 2008).

5.1 EXPERIMENTOS COM SETI@HOME E EINSTEIN@HOME

SETI (sigla em inglês para *search for extraterrestrial intelligence*, que significa Busca por Inteligência Extraterrestre) é um projeto que tem por objetivo analisar sinais de rádio captados por radiotelescópios com finalidade de identificar algum padrão inteligente vindo do espaço. SETI@home (Berkeley, University of California, 2014) é um projeto feito com base nas pesquisas do projeto SETI que utiliza os dados coletados por ele, dividindo-os em pequenos trechos que possam ser analisados por computadores pessoais comuns.

A aplicação SETI@home é executada em forma colaborativa de usuários pelo mundo, quem desejar ajudar no projeto pode baixar o software e executar na própria máquina, as tarefas são executadas e os resultados enviados ao servidor do projeto. Cada colaborador recebe pequenos pacotes de dados para análise.

É um software que aproveita bem a GPGPU e além disso utiliza Transformadas Rápidas de Fourier (FFT – *Fast Fourier Transform*), não comentado anteriormente mas que existe implementação especial em várias GPUs. O Projeto acaba também ajudando a mapear o céu mantendo uma base de dados mais rica de informações.

Einstein@home (UWM Physics Department, 2014) é bem similar ao SETI@home na forma colaborativa de usuários embora seu foco seja outro, ele procura por pulsares e ondas gravitacionais emitidas por pulsares, buracos negros, estrelas de nêutrons, estrelas de quarks e outros objetos bem densos, que, teoricamente, podem emitir fortes ondas gravitacionais, a pesquisa já descobriu mais de 30 estrelas de nêutrons. Os dados são distribuídos entre os computadores voluntários para análise e depois os resultados são enviados ao servidor do projeto. Também é uma aplicação que aproveita a GPGPU.

5.1.1 SETI@home com uma GeForce GT520

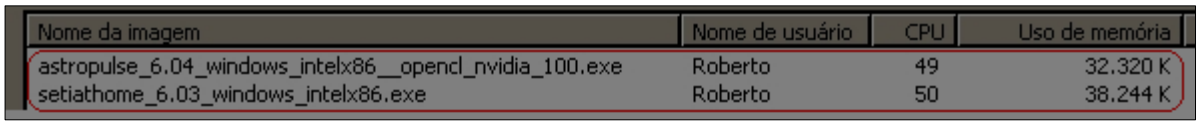
Especificação do computador neste experimento:

- Intel Core 2 Duo E5700 2,93 GHz - 3,5 GB RAM
- GPU Nvidia GeForce GT 520 (48 cores CUDA, 1 GB DDR3)
- Windows XP 32b com Service Pack 3

Foram comparados em torno de uma hora e meia após início de processamento, 2 (dois) pacotes de trabalho do projeto rodando ao mesmo tempo na máquina, um deles com o uso da GPGPU e o outro somente com recurso da CPU. Ambos competindo com o mesmo recurso de CPU:

Imagem 8

Concorrência na CPU entre 2 pacotes de trabalho do projeto SETI@home



Fonte: Próprio Autor, *print screen* parcial da tela do Gerenciador de Tarefas do Windows

Tabela 2.

Resultados obtidos de processamento no projeto SETI@home

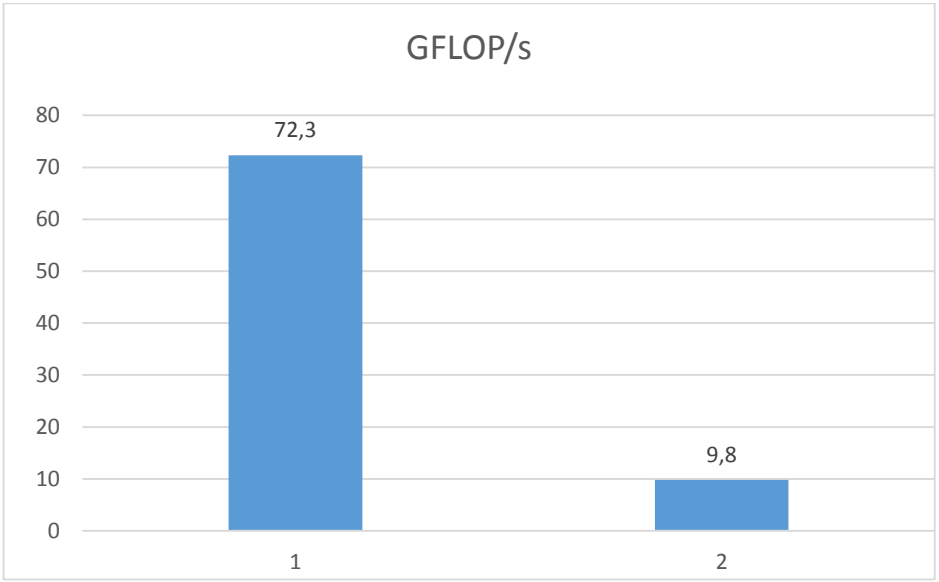
	Tarefa 1	Tarefa 2
GPGPU ?	SIM	NÃO
Tempo de Processamento	5673s	5443s
GFLOP Concluído	410.155	53.543
Velocidade (GFLOP/s)	72,3	9,8

Fonte: Próprio autor, resultados coletados através do monitor BOINC do SETI@home.

Embora ambas as tarefas estejam teoricamente com o mesmo recurso de CPU, aquela utilizando GPGPU está executando 7,3 vezes mais instruções por segundo.

Gráfico2

Comparação SETI@home GPGPU x CPU



(1) SETI@home com recursos de GPGPU (OpenCL),

(2) SETI@home somente com recursos de CPU.

Fonte: Experimento do Próprio autor.

5.1.2 Enstein@home com uma GeForce GT640

Especificação do computador neste experimento:

- Intel Dual Core E5700 3 GHz – 4 GB RAM
- GPU Nvidia GeForce GT 640 (384 cores CUDA, 2 GB DDR3)
- Windows 7 64b com Service Pack 1

Foram comparados 2 (dois) pacotes de trabalho do projeto rodando ao mesmo tempo na máquina, um deles com o uso da GPGPU e o outro somente com recurso da CPU. Ambos competindo com o mesmo recurso de CPU:

Tabela 3

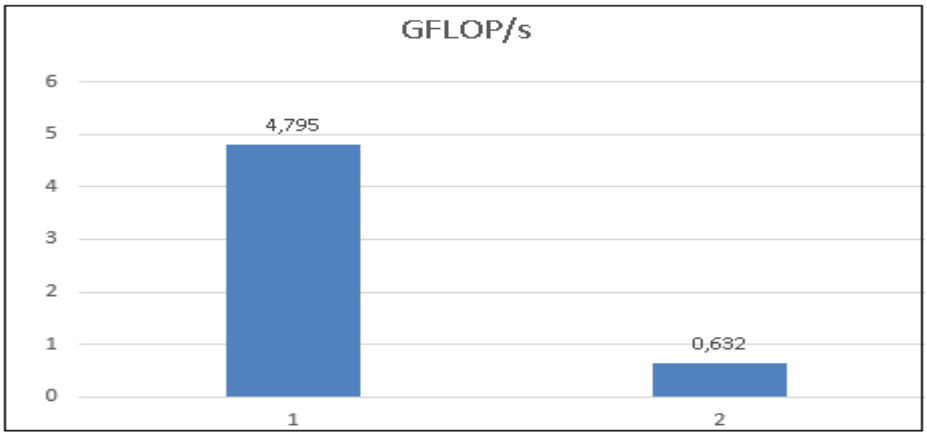
Resultados obtidos de processamento no projeto EINSTEIN@home

	Tarefa 1	Tarefa 2
GPGPU ?	SIM	NÃO
Tempo de Processamento	3817s	13282s
GFLOP Concluído	18.303	8.399,6
Velocidade (GFLOP/s)	4,79	0,63

Fonte: Próprio autor, os resultados foram coletados através do monitor BOINC do Einstein@home

Gráfico3

Comparação Einstein@home GPGPU x CPU



(1) Einstein@home com recursos de GPGPU (OpenCL),

(2) Einstein@home somente com recursos de CPU.

Fonte: Próprio Autor.

Embora ambas as tarefas estejam teoricamente com o mesmo recurso de CPU, aquela utilizando GPGPU está executando 7,5 vezes mais instruções por segundo.

5.2 EXPERIMENTOS COM O JOGO N-RAINHAS

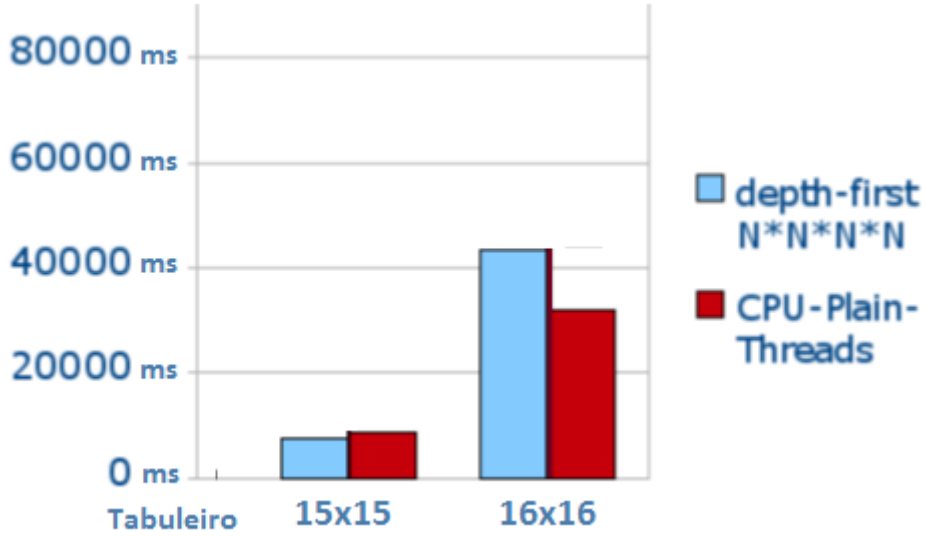
O jogo N-Rainhas é como o de Xadrez, possui características de soluções através de busca em árvores, algoritmos que caminham nos N nós de uma árvore em profundidade. Não é possível fazer o aproveitamento da alocação de um bloco completo da GPGPU, não é conhecida alguma forma de processar todos os nós ao mesmo tempo em uma matriz, porque o próximo nó só é conhecido após a avaliação do nó corrente. Apenas por essa avaliação, pode-se inferir que para este problema o uso da GPGPU não é favorável. Pode-se inferir também que se for usar GPGPU para este problema, quanto mais blocos da GPU usar melhor, mas ficariam ociosas as *threads* que poderiam ser executadas em paralelo dentro de um mesmo *warp*. Vejamos, executar apenas uma *thread* ou duas por *warp* e executar 4 blocos em paralelo significa a perda de pelo menos 30 *threads* por *warp* X 4, ficariam ociosas 120 threads por causa da característica do problema, então uma CPU de 4 núcleos deve se sair muito melhor para este caso, pois é mais rápida (comparando por exemplo uma CPU a 3GHz e um bloco da GPU a 400 MHz), mesmo que pensemos em pegar todos os blocos disponíveis da GPU (para uma placa GTX 580 seriam 32) para executar em paralelo pensando em uma similaridade com núcleos de uma CPU em paralelo, ainda assim é necessário contar que os núcleos da GPU trabalham com uma frequência menor e que há um gasto de tempo adicional para gerenciar a lógica de troca de dados entre as memórias da GPU dos SMs para a memória global e entre a GPU e a CPU. A menos que se descubra um outro meio de solucionar o jogo N-Rainhas, a hipótese é que uma solução por GPGPU não seja melhor do que a execução tradicional de *threads* através somente da CPU.

E para confirmar esta hipótese, foi comparado um trabalho já realizado a respeito por Vitor Fernando Pamplona (PAMPLONA, 2008). O objetivo foi executar os algoritmos *n-deph* para resolver o problema N-Rainhas através da CPU e depois da GPU e fazer uma comparação dos resultados. Pode-se observar que realmente para

este problema a GPGPU não foi mais eficaz que a execução tradicional em *threads* na CPU. O gráfico 4 mostra o tempo em milissegundos para serem encontradas as soluções considerando um tabuleiro 15X15 e outro 16x16.

Gráfico 4

N-Rainhas, GPGPU x CPU



Fonte: Adaptação (PAMPLONA, 2008)

Cabe ressaltar aqui que antes de grandes investimentos em GPUs para fins de acelerar o processamento, é necessário avaliar se o problema que se apresenta é favorável a soluções por GPGPU.

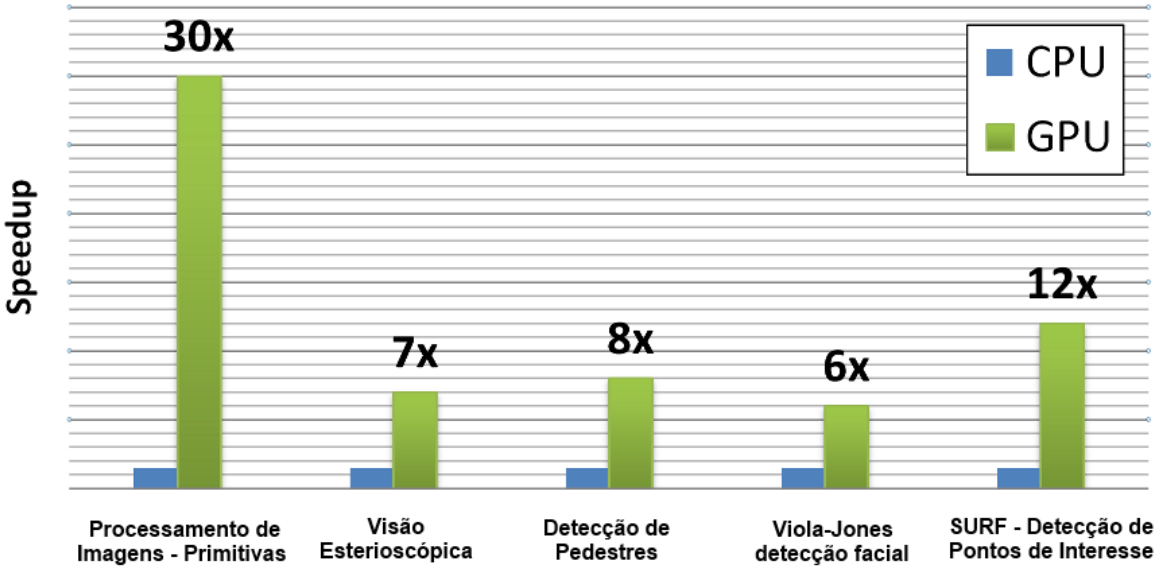
5.3 PERFORMANCE COM OpenCV

Como já visto em 2.4.5, OpenCV é uma API utilizada para objetivos de visão computacional e reconhecimento de imagens. Por sua característica de trabalho com imagens, vários algoritmos são adequados ao paralelismo através da GPU.

Abaixo está uma comparação realizada pelos próprios desenvolvedores da OpenCV. Neste caso foi avaliada uma GPU NVidia Tesla C2050 contra uma CPU Intel

i5-760 2,8GHz. Este modelo de GPU Tesla possui 448 CUDA Cores a 1.15 GHz e com memória dedicada GDDR5 de 3GB. A CPU Intel Core i5-760 é um processador de 4 (quatro) núcleos.

Gráfico 5
Performance com OpenCV



Fonte: OpenCV (2014), com tradução nossa.

Há ótimas possibilidades usando OpenCV e GPGPU em aplicativos que necessitam de interação do usuário e reconhecimento de imagens em tempo real.

Para melhor entendimento, a figura abaixo mostra a imagem de uma aplicação de Detecção de Pedestres.

Imagem 9

Detecção de Pedestres

Fonte: Pedestrian Detection (2014).

6 CONSIDERAÇÕES FINAIS

A programação usando GPGPU exige conhecimento sobre a arquitetura e uso dos Streaming Processors, além do uso de uma lógica diferente da lógica linear usada na programação de CPUs. Portanto podemos concluir que o custo é maior para o desenvolvimento de programas com GPGPU no que se refere a profissionais que atuam no projeto de software e na programação e também mais tempo paro que o produto final fique pronto.

A GPGPU é sensível ao problema que se apresenta. Há problemas apropriados a soluções por GPGPU e outros não. Considerando um problema apropriado, a GPGPU pode trazer grande economia no consumo de energia, espaço físico para máquinas, valor dos equipamentos e menor tempo de processamento.

Em geral as melhores aplicações para usar GPGPU são aquelas que envolvem visão computacional, processamento de imagens e problemas que envolvem cálculos que podem ser aplicados ao mesmo tempo sobre uma massa de dados.

A GPGPU está estreitamente relacionada com a supercomputação. Está presente em grandes pesquisas no mundo, simulações sísmicas, previsão do tempo e outros.

Também tem relação com profissionais de Jogos Digitais, mesmo que indiretamente. Está presente nas engines para games. Softwares com suporte a GPGPU permitem a execução de simulações mais rapidamente, e mecanismos que envolvem o reconhecimento por visão computacional permitem interações mais sofisticadas com os usuários. Possibilidades aumentam para imersão de jogadores em um mundo virtual.

Como atualmente há dispositivos móveis equipados com GPUs compatíveis, podemos dizer que é possível ter aplicações de RA e outros de visão computacional na palma da mão. Vários produtos construídos com OpenCV podem ser pensados para serem executados em dispositivos móveis.

Possibilidades de outros trabalhos relacionados:

- Implementação de aplicações que envolvem Visão Computacional em tempo real em dispositivos móveis dotados de GPU Tegra (Aplicações envolvendo RA,

identificação de marcadores virtuais, detecção de pedestres). Os dispositivos móveis capazes de executar GPGPU são recentes no mercado.

- Estudos práticos de algoritmos implementados para execução através de GPGPU em diferentes modelos de Placas de Vídeo. Como a quantidade de Streaming Processors varia conforme cada modelo de GPU, há de se adotar alguma técnica para melhor aproveitamento de cada SM.

- Investigação da GPGPU para novos produtos que estão aparecendo no mercado. O Google Glass por exemplo, atualmente usa a GPU PowerVR SGX 540 da Imagination Technologies mas pouco se sabe sobre o suporte com OpenCL. Pode-se deduzir que seria muito bom existirem aplicativos que utilizam OpenCV operando com o Google Glass, qual seria então a comparação de uma aplicação assim operando no Google Glass e em outro dispositivo móvel com GPU?

- Investigação a respeito dos recentes Coprocessadores Xeon Phi da Intel. Da mesma forma que modernas GPUs, as placas Xeon Phi (imagem abaixo) podem ser adicionadas ao computador através das interfaces PCI Express, uma ou várias delas e diferentemente das GPUs, é um produto que já nasceu para finalidades de programação paralela.

Imagem 10

Coprocessador Xeon Phi

Fonte: Intel (2014)

7 REFERÊNCIAS BIBLIOGRÁFICAS

Berkeley, University of California. **SETI@home**. <http:// setiathome.ssl.berkeley.edu/> Acesso em: 08/06/2014.

BLITZ Research. **About Blitz3D**. Disponível em: <http:// www.blitzbasic.com/ Products/ blitz3d.php> Acesso em: 08/06/2014.

EPIC GAMES. **About Unreal Engine**. Disponível em: <https:// www.unrealengine.com/ products/ unreal-engine-4> Acesso em: 08/06/2014.

Fantalgo. **General Purpose GPU (GPGPU) Computing**. Disponível em: <http:// www.fantalgo.com/ GPGPU.html> Acesso em 08/06/2014.

INTEL. **Intel® Xeon Phi™ Product Family Overview**. Disponível em: <http:// www.intel.com/ content/ www/ us/ en/ processors/ xeon/ xeon-phi-architecture-for-discovery-video.html> Acesso em: 25/05/2014.

ISTOÉ Dinheiro. **Um supercomputador de mesa**. Ed. Digital 719 de 15/07/2011. Disponível em: <http:// www.istoedinheiro.com.br/ noticias/ mercado-digital/ 20110720/ supercomputador-mesa/ 2977.shtml> Acesso em: 08/02/2013.

Khronos Group (2014a). **OpenGL Overview**. Disponível em: <http:// www.opengl.org/ about/> Acesso em: 08/06/2014.

Khronos Group (2014b). **OpenCL – The open standard form parallel programming of heterogeneous systems**. Disponível em: <http:// www.khronos.org/ opencl/> Acesso em 08/06/2014.

MICROSOFT. **Introdução ao desenvolvimento de jogos DirectX**. Disponível em: <http:// msdn.microsoft.com/ pt-br/ library/ windows/ apps/ jj554502> Acesso em: 08/06/2014.

NVIDIA Corporation (2014a). **Plataforma de Computação Paralela, Cuda Zone**. Disponível em: <http:// www.nvidia.com.br/ object/ cuda_home_new_br.html> Acesso em: 08/06/2014.

NVIDIA Corporation (2014b). **O que é computação com GPU**. Disponível em: <http:// www.nvidia.com.br/ object/ old-what-is-gpu-computing-br.html> Acesso em: 31/05/2014.

NVIDIA Corporation (2014c). **NVIDIA Developer Zone, GPU-ACCELERATED LIBRARIES**. Disponível em: <https:// developer.nvidia.com/ gpu-accelerated-libraries> Acesso em 08/06/2014.

NVIDIA Corporation (2014d). White Paper: NVIDIA's Next Generation, **CUDA™ Compute Architecture: Fermi™**. V1.1. Disponível em: <http:// www.nvidia.com.br/ content/ PDF/ fermi_white_papers/ NVIDIA_Fermi_Compute_Architecture_ Whitepaper.pdf> Acesso em: 23/05/2014.

NVIDIA Corporation (2014e). **ABOUT CUDA**. Disponível em: <https:// developer.nvidia.com/ about-cuda> Acesso em: 08/06/2014.

NVIDIA Corporation (2014f). CUDA Toolkit Documentation - **Programming Model**. Disponível em: <http:// docs.nvidia.com/ cuda/ cuda-c-programming-guide/ index.html#programming-model> Acesso em: 08/06/2014.

OpenCV. **Platforms**. Disponível em <http:// opencv.org/ platforms.html> Acesso em: 22/05/2014.

PAMPLONA, Vitor Fernando. **N-Rainhas: Comparação entre CPU e GPU usando C++ e Cuda**. UFRGS - Instituto de Informática, 2008.

Pedestrian Detection. **Home**. Disponível em: <http:// www.pedestrian-detection.com/> Acesso em: 23/05/2014.

PETROBRAS. Fatos e Dados. **Conheça nosso supercomputador que gera mais de seis trilhões de amostras sísmicas por segundo**. Disponível em <http:// www.petrobras.com.br/ fatos-e-dados/ conheca-nosso-supercomputador-que-gera-mais-de-seis-trilhoes-de-amostras-sismicas-por-segundo.htm> Acesso em: 24/05/2014.

SEESP - Sindicado dos Engenheiros no Estado de São Paulo. **PETROBRAS muda tecnologia e monta supercomputador**. Disponível em <http:// www.seesp.org.br/ site/ cotidiano/ 1796-petrobras-muda-tecnologia-e-monta-supercomputador.htm> Acesso em: 24/05/2014.

Sundog Software. **3D Oceans with Triton**. Disponível em: <http:// sundog-soft.com/ sds/ features/ ocean-and-water-rendering-with-triton/> Acesso em: 10/03/2013.

TARIQ, Sarah. Presentation: **An Introdution to GPU Computing and CUDA Architecture**. In: NVIDIA Corporation, GTC EXPRESS WEBNARS, p. 10-12, jun. 2011. Disponível em: <http:// on-demand.gputechconf.com/ gtc-express/ 2011/ presentations/ GTC_Express_Sarah_Tariq_June2011.pdf> Acesso em: 14/02/2013.

Unity Technologies. **Create the games you love with Unity**. <http:// unity3d.com/ unity> Acesso em: 08/06/2014.

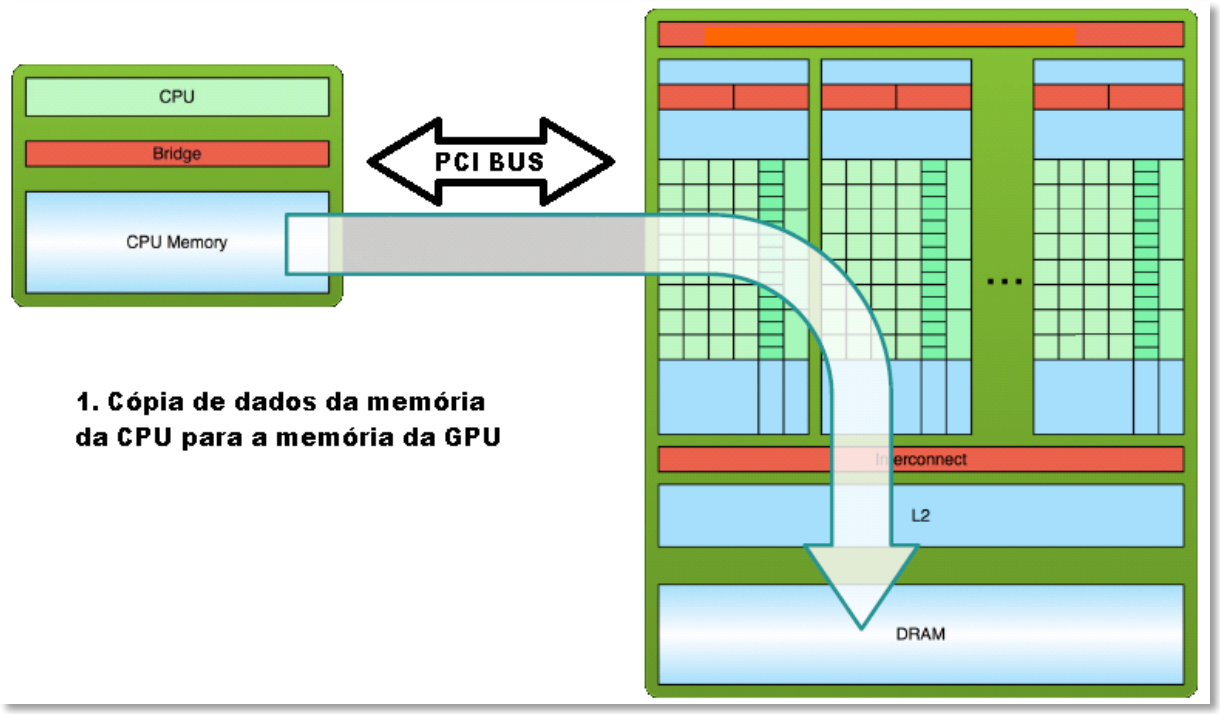
UWM Physics Department. **About Einstein@Home**. Disponível em: <http:// einstein.phys.uwm.edu/> Acesso em: 08/06/2014.

WIKIPEDIA. **DirectX**. Disponível em: <http:// pt.wikipedia.org/ wiki/ DirectX> Acesso em: 08/06/2014.

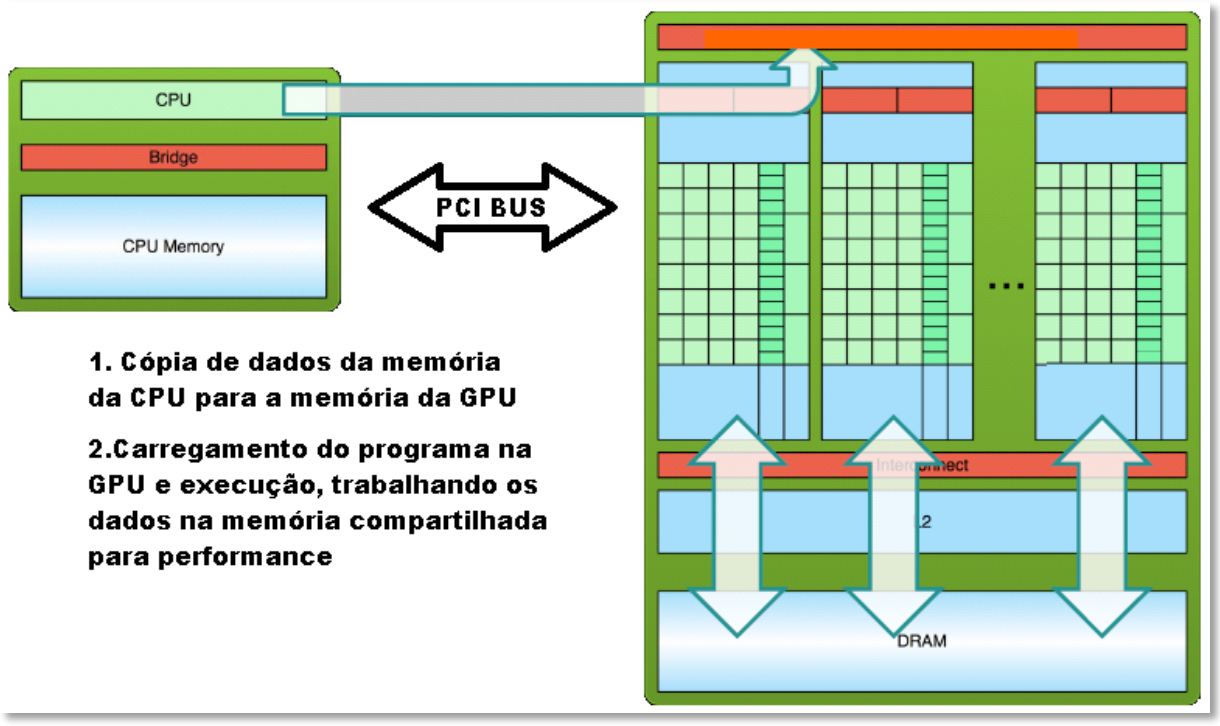
8 APÊNDICE 1

FLUXO SIMPLIFICADO DE PROCESSAMENTO EM GPGPU

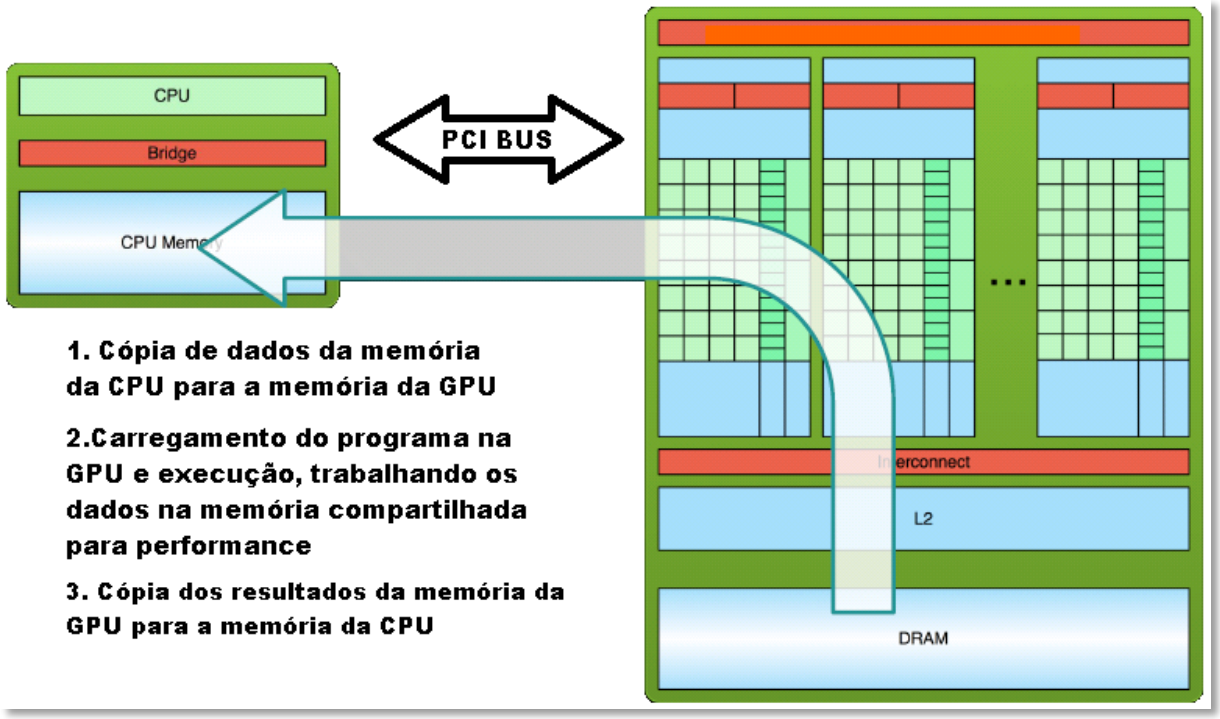
Passo 1



Passo 2



Passo 3



Imagens adaptadas para o Português da apresentação original "*An Introdution to GPU Computing and CUDA Architecture*" (TARIQ, 2011).