Introdução à Arquitetura de GPUs

Marcos Vinícius Mussel Cirne
RA: 045116
Instituto de Computação - IC / UNICAMP
marcosvcirne@gmail.com

RESUMO

Este relatório fará uma abordagem sobre o funcionamento das GPUs, explorando-se os conceitos básicos, suas aplicações e alguns detalhes sobre as suas arquiteturas. Além disso, será mostrado todo o histórico da evolução destas unidades, desde os anos 70 até os dias atuais, bem como os principais fatores que fizeram com que as GPUs evoluíssem rapidamente ao longo dos anos, obtendo um crescimento exponencial na sua performance geral.

1. INTRODUÇÃO

As GPUs (do inglês *Graphics Processing Unit*) são processadores especializados que basicamente efetuam operações ligadas a aplicativos gráficos 3D. São usadas com frequência em sistemas embarcados, jogos, consoles de videogames, estações de trabalho, etc. Elas também são bastante eficientes na manipulação de tarefas de Computação Gráfica em geral, conseguindo obter um desempenho superior ao de CPUs de propósito geral, devido à sua estrutura altamente paralelizada. Em computadores pessoais, elas podem estar presentes nas placas de vídeo ou então integradas à placa-mãe (tal como ocorre nos notebooks).

Basicamente, as GPUs são dedicadas para o cálculo de operações de ponto flutuante, destinadas a funções gráficas em geral. Os algoritmos de renderização normalmente trabalham com uma quantidade significativa dessas operações, as quais são executadas de uma maneira eficiente, graças ao uso de microchips próprios que contém um conjunto de operações matemáticas especiais. O bom desempenho desses microchips garante a eficiência das GPUs de uma forma geral.

Outras operações implementadas nas GPUs são aquelas que lidam com operações primitivas, como traçado de retas e círculos, desenhos de triângulos, retângulos, etc. GPUs mais modernas também fazem uso de operações relacionadas a vídeos digitais, bastante úteis para a Computação Gráfica em 3D.

Atualmente, existem várias empresas que atuam no mercado de GPUs pelo mundo. As duas principais são a NVIDIA, com a linha GeForce, e a ATI, com a linha Radeon. Além destas, a Intel também se destaca no ramo, mas as suas GPUs geralmente são fabricadas de forma integrada.

2. HISTÓRICO

Nos anos 70, foram criados os primeiros computadores da família Atari 8-bits (também chamados de *home computers*). Eles eram fabricados com chips ANTIC, que processavam instruções de display e eram responsáveis pelo "background" das telas gráficas, e CTIA, presentes nos modelos Atari 400 e 800 e realizavam operações de adição de cores e sprites[1] aos dados processados pelos chips ANTIC. Os chips CTIA foram posteriormente substituídos pelos chips GTIA nos modelos seguintes.

Nos anos 80, o Commodore Amiga foi um dos primeiros computadores de produção em massa a incluir um blitter[2] em seu hardware de vídeo. Além dele, o sistema 8514 da IBM foi uma das primeiras placas de vídeo para PC que implementavam primitivas 2D em hardware. Nessa época, o Amiga era considerado um acelerador gráfico completo, justamente pelo fato de ele transferir todas as funções alusivas à geração de vídeos para hardware. No entanto, uma CPU de propósito geral era necessária para lidar com todos os aspectos de desenho do display.

Já nos anos 90, continuava-se a evolução dos aceleradores 2D. Uma vez que as técnicas de fabricação se tornavam cada vez mais avançadas, a integração de chips gráficos seguia os mesmos passos. Foram surgindo uma série de API's (*Application Programming Interfaces*) que lidavam com um conjunto de tarefas. Enrte eles, estavam a biblioteca gráfica *WinG*, que aumentava a velocidade e a performance dos ambientes para o Windows 3.x, além de fazer com que jogos desenvolvidos para DOS fossem portabilizados para a plataforma Windows, e também o *DirectDraw*, que provia aceleração de hardware para jogos 2D em Windows 95 e nas versões posteriores.

Alguns anos mais tarde, a renderização de gráficos 3D em tempo real era bastante comum em jogos de computador e de alguns consoles, o que exigiu uma demanda cada vez maior

[1] Imagens ou animações em 2D / 3D que são integradas em uma cena maior.
[2] Dispositivo que realiza rápidas transferências de uma área de memória para outra.

Figura 1: Placa de vídeo 3dfx Voodoo3.

de placas gráficas aceleradoras 3D. Era o auge dos chamados consoles de videogames da quinta geração, tais como o PlayStation e o Nintendo 64. No tocante aos computadores, surgiram alguns chips gráficos 3D de baixo custo, como o S3 Virge, o ATI Rage e o Matrox Mystique, mas nenhum deles obteve muito sucesso. Isso se deve ao fato de esses chips serem basicamente aceleradores 2D com algumas funções de geração de gráficos 3D embutidas. No início, a performance obtida com gráficos 3D era obtida somente por meio de algumas placas dedicadas a funções de aceleração 3D, mas tendo uma baixa aceleração 2D como um todo. Uma dessas placas era a *3dfx Voodoo*, a primeira placa gráfica comercial, fabricada em 1995, e que era capaz de realizar mapeamento de textura em geometrias e possuía o algoritmo de Z-Buffer[3] implementado em hardware.

Em 1999, a NVIDIA lançou o modelo GeForce 256, considerada a primeira placa gráfica do mundo. Ela se destacou na época pelo alto número de pipelines, pela realização de cálculos de iluminação e de geometria e pela adição de recursos para compensação de movimentos de vídeo MPEG-2. Além disso, possibilitou um avanço considerável no desempenho em jogos e foi o primeiro acelerador gráfico compatível com o padrão DirectX 7.0. O sucesso desta placa foi tanto que causou a queda de seus concorrentes diretos no mercado, entre eles a 3dfx, fabricante da Voodoo3.

A partir de 2000, as GPUs incorporavam técnicas de *pixel shading*, onde cada pixel poderia ser processado por um pequeno programa que incluía texturas adicionais, o que era feito de maneira similar com vértices geométricos, antes mesmo de serem projetados na tela. A NVIDIA foi a primeira a produzir plcasa com tais características, com o modelo GeForce 3. Em meados de 2002, a ATI introduziu o modelo Radeon 9700, que foi a primeira placa do mundo compatível com o acelerador DirectX 9.0.

Em 2005, foi criado o barramento PCIe, que melhorava ainda mais o desempenho das transferências de dados entre GPU e CPU. A partir dele, foram criadas as tecnologias SLI, da NVIDIA, e CrossFire, da ATI, as quais permitiam interligar múltiplas GPUs em um único sistema, o que se assemelha à concepção de CPUs com múltiplos núcleos (cores).

Mesmo com toda essa evolução, uma vez que o poder de processamento das GPUs aumentava ao longo dos anos, elas exigiam cada vez mais energia elétrica. As GPUs da alta

[3]Gerenciamento de coordenadas de imagem em 3D. Muito utilizado para a solução de problemas de visibilidade de objetos em uma cena. Também conhecido como *depth buffering*.

Figura 2: Placa de vídeo ATI Radeon HD 4870.

Figura 3: Placa de vídeo GeForce GTX 295.

performance normalmente consomem mais energia do que as CPUs atuais.

Nos dias de hoje, as GPUs paralelas tem disputado espaço com as CPUs, principalmente com a criação de técnicas como a GPGPU (*General Purpose Computing on GPU*), que consiste basicamente em fazer com que as GPUs executem tarefas de alto processamento gráfico, o que até certo tempo atrás era realizado pelas CPUs. Essa técnica possui diversas aplicações em processamento de imagens, álgebra linear, reconstrução 3D, entre outras áreas. Existem também uma forte demanda por parte dos adeptos da GPGPU para melhorias no design do hardware, com o intuito de tornar os modelos de programação mais flexíveis.

3. PIPELINES GRÁFICOS

Todos os dados processados pelas GPUs seguem o que se chama de pipeline gráfico, desenvolvido para manter uma alta frequência de computação por meio de execuções paralelas. O pipeline gráfico convencional, mostrado na figura 4, é composto por uma série de estágios, executados através de parâmetros definidos por uma API associada. Inicialmente, a aplicação envia à GPU, via barramento, um conjunto de vértices a serem processados. Em seguida, são definidas as primitivas geométricas utilizadas para o processo de rasterização, o qual consiste na conversão dessas primitivas em conjuntos de pixels. Em seguida, a GPU faz uma combinação dos vértices para gerar fragmentos para cada um dos pixels da imagem de saída, onde cada fragmento é definido por um conjunto de operações de rasterização, que incluem mapeamento de textura, combinação de cores, recortes, etc. O resultado dessa operação é então encaminhado ao framebuffer, também conhecido como memória de vídeo, que se encarrega de armazenar e transferir para a tela os dados processados nos estágios anteriores do pipeline.

Com o surgimento das placas gráficas programáveis, houve também uma mudança na concepção geral do pipeline gráfico. Alguns estágios do pipeline gráfico podem ser substituídos por programas que manipulam vértices e fragmentos. No entanto, quando um programa desse tipo é implemen-

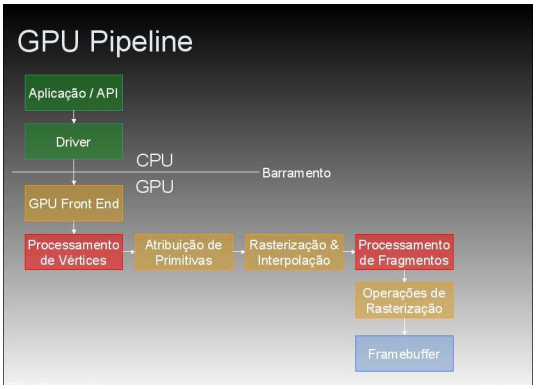

Figura 4: Esquema geral do pipeline gráfico.

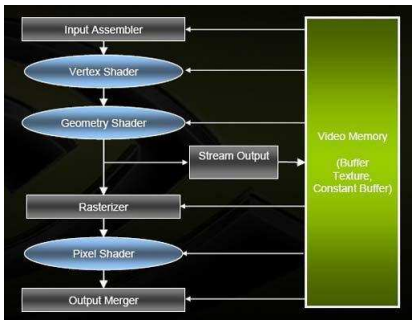

Figura 5: Novo pipeline gráfico.

tado, ele deve também implementar todos os estágios do pipeline gráfico que ele substitui.

Além disso, um novo estágio foi adicionado ao pipeline gráfico programável: o estágio de geometria, responsável por todo o processo de renderização de uma cena. Dessa forma, o novo pipeline adquire os estágios de *vertex shader*, *geometry shader* e *pixel shader*. O primeiro shader é normalmente usado para adicionar efeitos especiais em objetos presentes em um ambiente 3D. O segundo serve para gerar novas primitivas geométricas a partir dos dados recebidos do vertex shader. E o último tem a função de adicionar efeitos de luz e sombreamento aos pixels de uma imagem 3D.

A adição do estágio de geometria divide o pipeline em duas partes, como mostrado na figura 5. Um deles é o estágio de *Stream-Output*, responsável pela alocação de vértices oriundos do geometry shader em um ou mais buffers na memória, podendo também gerar novos vértices para o pipeline. O outro consiste no processo de rasterização descrito anteriormente, mas agora gerando uma entrada para o pixel shader.

Na época de lançamento do DirectX 10, as GPUs eram implementadas de forma a atender aos seus requisitos. Com isso, foi introduzida uma nova unidade, definida como *unified shader*, inicialmente introduzido no chip GeForce 8800 GTX, e que agrega todos os outros shadres citados anteriormente, conforme exibido na figura 6.

Recentemente, foi especulado um novo pipeline gráfico programável com a inclusão de novos shaders, graças à imple-

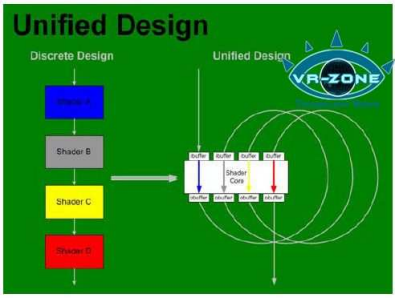

Figura 6: Agregação dos shaders em uma única unidade: o *unified shader*.

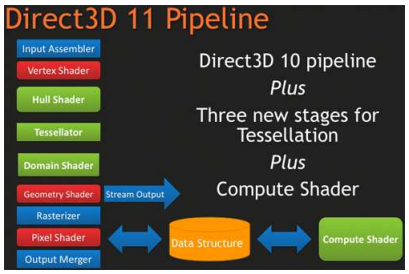

Figura 7: Pipeline do Direct3D 11, com a inclusão de novos shaders (e consequentemente, novos estágios).

mentação do Direct3D 11, que provavelmente será lançado no segundo semestre de 2009. O esquema desse pipeline está representado na figura 7.

4. CARACTERÍSTICAS DO HARDWARE

As GPUs foram projetadas com a ideia de executar todo tipo de operação gráfica. Normalmente, na execução de aplicações gráficas, uma mesma operação é feita repetidas vezes com vários dados diferentes, fazendo com que a arquitetura das GPUs fossem desenvolvidas para operar grandes conjuntos de instruções de maneira paralela. Para tornar esse paralelismo eficiente, o hardware foi implementado com base nos pipelines gráficos, conforme descrito na seção 3. Os pipelines, por sua vez, foram então implementados em várias unidades independentes. Desta forma, as GPUs conseguem manter altas frequências de computação em execuções paralelas de instruções.

Outra característica importante a ser ressaltada é o fato de o hardware das GPUs ser especializado, obtendo uma eficiência muito maior do que os hardwares de propósito geral (como as CPUs). Ao longo dos anos, a disparidade entre GPUs e CPUs, em relação à performance, foi se tornando cada vez maior, como mostra o gráfico da figura 8.

Uma segunda vantagem que as GPUs levam sobre as CPUs é que elas utilizam a maioria de seus transistores para a computação e muito pouco para a parte de controle, obtendo-se um poder de computação maior, mas tornando os fluxos de programas um pouco mais limitados.

Outro aspecto importante a se considerar é com relação ao acesso à memória. As GPUs procuram maximizar o th-

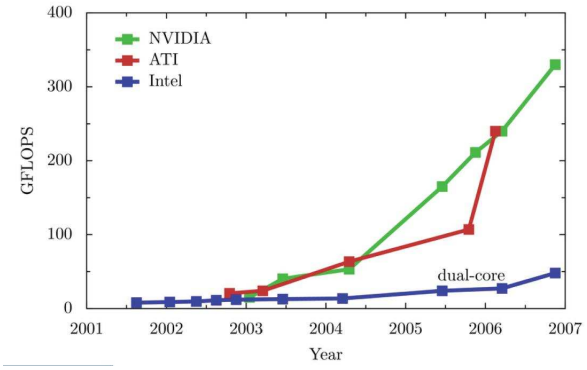

Figura 8: Evolução das performances dos processadores Intel e das placas de vídeo ATI e GeForce.

roughput, ao passo que as CPUs priorizam a latência no acesso. Isso ocorre porque a computação das GPUs tira mais proveito do chamado princípio da localidade do que a computação genérica. Nesse caso, valoriza-se mais a eficiência na transferência de um determinado conjunto de elementos, em detrimento do tempo gasto para acessá-los.

Atualmente, a GPU é o tipo de hardware mais barato para a computação de alto desempenho, quando a tarefa a ser processada se aproveita do seu alto grau de paralelismo. No entanto, mesmo que ela esteja se tornando cada vez mais programável, ainda é necessário um conjunto de fatores que proporcionem a melhor performance possível para uma determinada aplicação.

Dada essa eficiência das GPUs, podemos então imaginar que uma aplicação que é executada em uma CPU normal tem uma performance melhor quando executada em uma GPU. Esta é uma concepção leviana, já que nem toda aplicação pode ser diretamente adaptada para o uso em uma GPU, uma vez que é necessária a remodelagem para um problema gráfico, o que não é possível em boa parte dos casos.

5. MODELO DE PROGRAMAÇÃO

Um dos motivos pelos quais as CPUs não conseguem uma eficiência tão boa quanto as GPUs está no modelo de programação utilizado. As CPUs são desenvolvidas sob um modelo de programação serial, o qual não explora muito o conceito de paralelismo e os padrões de comunicação nas aplicações. Em contrapartida, as GPUs utilizam, como base, o modelo de programação por streams[4], que estrutura os programas de uma forma que se obtenha uma alta eficiência na computação e na comunicação.

No modelo de programação por streams, os dados podem ser simples (inteiros, pontos flutuantes) ou complexos (pontos, triângulos, matrizes de transformação). As streams também podem ter um tamanho qualquer, sendo que as operações executadas sobre as streams são mais eficientes se elas forem longas (contendo centenas de elementos ou mais). As operações que podem ser feitas nas streams são as de cópia de streams, construção de substreams a partir delas, indexação e computação por meio de kernels[5].

A eficiência na computação proveniente deste modelo está no alto grau de paralelismo dos dados na aplicação, onde os elementos das streams podem ser processados a partir de um hardware especializado em tal aspecto. Dentro de cada elemento processado, pode-se explorar um paralelismo a nível de instrução (ILP). Uma vez que as aplicações são constituídas de múltiplos kernels, eles também podem ser processados em paralelo, utilizando-se um paralelismo a nível de tarefa (TLP).

A adoção deste modelo no desenvolvimento de GPUs reflete uma série de tendências. A primeira delas é a habilidade de se concentrar uma vasta quantidade de dados a serem computados em um simples *die* do processador, bem como o uso eficiente desses componentes. Outra tendência está associada à queda no custo da produção de GPUs, fazendo com que elas se tornassem uma parte de um desktop padrão. Por fim, temos a adição de uma programabilidade de alta precisão ao pipeline gráfico, completando-se uma transição de um simples processador especializado para um poderoso processador programável que pode desempenhar uma grande variedade de tarefas.

6. DESAFIOS FUTUROS

O primeiro desafio a se considerar diz respeito às tendências da tecnologia. Cada nova geração de hardware se tornará um novo desafio para os fabricantes de GPUs, pois nesse caso é necessário integrar novos recursos de hardware de maneira eficiente, com o objetivo de melhorar a performance e a funcionalidade. Um número cada vez maior de transistores é adicionado às placas gráficas, proporcionando maior exploração de paralelismo e novas funcionalidades ao pipeline gráfico.

Além disso, também é necessário garantir a eficiência na comunicação. O aumento do custo desse fator influencia nas microarquiteturas dos futuros chips. Desta forma, os desenvolvedores devem planejar o tempo exigido para o envio de dados ao longo dos chips.

Outro desafio para a evolução das GPUs é com relação ao consumo de energia, um fator que tem se tornado bastante crítico no desenvolvimento das GPUs atuais, uma vez que cada nova geração de hardware apresenta uma demanda de energia cada vez maior. É esperado então um maior gerenciamento de energia nos estágios do pipeline, bem como sistemas de resfriamento (cooling) ainda mais sofisticados.

Por fim, temos a questão da programabilidade e das funcionalidades que as GPUs oferecem. Mesmo com todo o progresso que se conquistou até os dias de hoje, a programabilidade em GPUs ainda está um pouco longe do ideal. Uma ideia para a evolução destes fatores é melhorar a flexibilidade e as funcionalidades das unidades programáveis atuais (de vértices e de fragmentos). Possivelmente, os seus conjuntos de instruções podem se convergir, tornando os seus controles de fluxos mais generalizados. Podemos ver também um

[4]Conjuntos ordenados de dados do mesmo tipo.

[5]Operador que age sobre streams, produzindo um outro conjunto de streams como saída. No caso do pipeline gráfico, deve-se implementar kernels para o processamento de vértices, de primitivas geométricas, e assim por diante.

compartilhamento de hardware programável entre esses dois estágios, de forma a utilizar melhor os seus recursos. Outra opção é expandir a programabilidade para unidades diferentes.

7. CONCLUSÃO

Mesmo sendo a GPU um componente recente no mundo da computação, ela vem mantendo uma evolução cada vez mais rápida e se tornando a cada dia mais útil para diversas aplicações gráficas. Isso pode ser evidenciado pela quantidade de modificações que ela sofre entre uma versão e outra, exigindo novas modelagens de seus componentes (ou até mesmo sobre as tecnologias utilizadas) e também novas implementações, visando melhoras ainda mais significativas na sua performance.

A introdução do conceito de programação genérica em placas gráficas (GPGPU) foi uma grande revolução para o uso das GPUs. Fabricantes de GPUs vem investindo bastante em melhorias na arquitetura de modo a tornar a programação cada vez mais simples e acessível aos programadores convencionais. Assim, esse conceito pode prover um ganho na eficiência do processamento de programas que não constituem essencialmente um problema gráfico.

É importante ressaltar também que essa evolução das GPUs, incorporando cada vez mais poder de processamento e capacidade nos seus chips, pode provocar algum conflito entre os seus fabricantes e os fabricantes de CPUs. Futuramente, não se sabe se as próximas gerações de computadores constituirão de CPUs que incorporarão funcionalidades básicas das GPUs ou então o contrário. Questões como esta são um bom desafio para os futuros projetistas, dentre uma série de outros desafios inerentes ao ramo tecnológico.

8. REFERÊNCIAS

[1] NVIDIA Corporation. Geforce256 – The World's First GPU, 1999.

[2] Randy Fernando, GPGPU: General-Purpose Computation on GPUs, 2004.

[3] Randy Fernando & Cyril Zeller, Programming Graphics Hardware, 2004.

[4] Artigo da Wikipedia – Graphics Processing Unit.

[5] NVIDIA Developer Zone – CPU Gems 2

[6] David Luebke (NVIDIA Research) & Greg Humphreys (University of Virginia), How GPUs work – artigo publicado em 2007, no site Computer Magazine, da IEEE Computer Society.