GPU sob o ponto de vista de arquiteturas paralelas, organização interna e utilização em sistemas de paralelismo massivo

Elisa de Cássia Silva Rodrigues[1]

[1]Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, Brasil.

Resumo. *Inicialmente projetadas para processamento de gráficos, as GPUs (*Graphics Processing Unit*) continham apenas funções de renderização fixas. Por serem processadores paralelos com alto poder computacional para cálculos aritméticos, as GPUs vêm evoluindo rapidamente com a inclusão de* pipelines *programáveis. A tendência é que a GPU se torne um processador genérico (GPGPU). Além disso, a integração entre núcleos GPU e CPU dentro um único chip também tende a se tornar um padrão entre os processadores. As GPUs também estão se tornando alvo de sistemas de paralelismo massivo como os supercomputadores. Hoje, o supercomputador mais rápido do mundo conta com milhares de GPUs, o que aumentou significativamente o seu desempenho com relação a seus antecessessores.*

1. Introdução

A CPU (*Central Processing Unit*) é a unidade de processamento central do computador. A maioria das CPUs *multicore* atuais usam o paradigma de memória compartilhada para comunicação com sincronização através de cache compartilhada. Cada núcleo tem uma *thread* de processamento por vez, com um conjunto de registradores contendo o estado da *thread*, uma ULA (Unidade Lógica Aritmética) dedicada para a *thread* atual e uma unidade grande dedicada ao gerenciamento e escalonamento de tarefas.

A GPU (*Graphics Processing Unit*) é uma unidade de processamento especializado em renderização de gráficos 3D. É usada em PCs, videogames e atualmente, também em dispositivos móveis como o iPhone da Apple. Inicialmente, as GPUs tinham o objetivo de processar apenas gráficos e processamento com ponto flutuante, o que justifica o fato de serem processadores bastante eficientes na manipulação de gráficos e cálculos matemáticos complexos. Essa eficiência deve-se a sua facilidade de processar vetores ou matrizes já que possuem mais circuitos para processamento do que as CPUs, que por sua vez possuem mais cache e controle de fluxo.

Uma característica marcante de uma GPU é a capacidade de processar trechos de código em paralelo de maneira muito eficiente, pois, enquanto as CPUs dedicam uma grande quantidade de seus circuitos ao controle, uma GPU foca mais nas unidades aritméticas. Tal poder de processamento paralelo torna a execução de algoritmos complexos mais eficiente, principalmente quando existem muitos dados a serem processados.

A evolução contínua das GPUs motivada principalmente pela necessidade de características como visualização de gráficos complexos, processamento de grande quantidade de cálculos numéricos e flexibilidade de programação, tem provocado uma discussão e incentivado cada vez mais a completa substituição das CPUs tradicionais por GPGPUs, ou seja, unidades de processamento gráfico de próposito geral.

Atualmente, as tecnologias ainda não possuem eficiência garantida quando considerado o processamento genérico. Questões como alto gasto de energia e dificuldade de programação de *software* para esses processadores são desafios a serem enfrentados. Grandes empresas envolvidadas no desenvolvimento dessas tecnologias (NVIDIA, Intel e AMD) disputam a imposição de seus padrões no mercado. Com isso, o investimento é cada vez maior gerando benefícios tanto para profissionais de TI quanto para simples usuários.

A tendência é a evolução das GPUs para as GPGPUs, um componente altamente paralelizável responsável tanto pelo processamento central quanto pelo processamento gráfico. Além disso, atualmente o grande foco das empresas de desenvolvimento dessas tecnologias está na integração e na colaboração entre CPU e GPU, a fim de aproveitar todo o poder de processamento das GPUs. As GPUs também estão se tornando alvo de sistemas de paralelismo massivo como os supercomputadores. Hoje, o supercomputador mais rápido do mundo conta com milhares de GPUs, o que aumentou significativamente o seu desempenho com relação a seus antecessessores.

1.1. Evolução da GPU

Entre as décadas de 1970 e 1980, surgiram as primeiras GPUS com funções gráficas fixas bem básicas, como desenhos de caracteres, linhas e arcos. A partir de 1990 houve um grande avanço no processamento gráfico com a evolução das GPUs que passaram a conter *pipelines* gráficas, chamadas de *shaders*, como a projeção de imagens, combinação de texturas e renderização 3D. A evolução desse hardware proporcionou a inclusão de novas *pipelines* programáveis e o suporte a ponto flutuante de precisão simples, no ínicio da década de 2000 [Kirk and Hwu 2010].

Com o avanço significativo de seu desempenho, as GPUs passaram a ser um grande atrativo na área de processadores dando suporte a ponto flutuante de precisão dupla e a várias linhas de código, surgindo então o conceito de *GPGPU (General-Purpose Computing on Graphics Processing Units).*

A Figura 1 apresenta a evolução das GPUs da NVIDIA apresentando características significativa incorporadas em cada geração. Neste ano, a NVIDIA lançou uma nova arquitetura, a GPU Kepler, uma das arquiteturas de computação de alto desempenho mais rápidas do mundo.

Atualmente, a utilização de GPGPU tem como principal vantagem possuir processadores de *threads* altamente paralelizáveis, o que favorece a execução de várias tarefas ao mesmo tempo. Outra vantagem é seu hardware de alta desempenho com um custo-benefício ótimo. No entanto, desenvolver *softwares* capazes de obter um ganho considerável para execução nesse hardware é um desafio pois devem respeitar algumas restrições como possuir estruturas de repetição aninhadas e paralelizáveis, alta computabilidade, acesso regular a dados e isolamento de dados entre CPU e GPU.

Table 1. NVIDIA GPU technology development.

Date	Product	Transistors	CUDA cores	Technology
1997	RIVA 128	3 million	—	3D graphics accelerator
1999	GeForce 256	25 million	—	First GPU, programmed with DX7 and OpenGL
2001	GeForce 3	60 million	—	First programmable shader GPU, programmed with DX8 and OpenGL
2002	GeForce FX	125 million	—	32-bit floating-point (FP) programmable GPU with Cg programs, DX9, and OpenGL
2004	GeForce 6800	222 million	—	32-bit FP programmable scalable GPU, GPGPU Cg programs, DX9, and OpenGL
2006	GeForce 8800	681 million	128	First unified graphics and computing GPU, programmed in C with CUDA
2007	Tesla T8, C870	681 million	128	First GPU computing system programmed in C with CUDA
2008	GeForce GTX 280	1.4 billion	240	Unified graphics and computing GPU, IEEE FP, CUDA C, OpenCL, and DirectCompute
2008	Tesla T10, S1070	1.4 billion	240	GPU computing clusters, 64-bit IEEE FP, 4-Gbyte memory, CUDA C, and OpenCL
2009	Fermi	3.0 billion	512	GPU computing architecture, IEEE 754-2008 FP, 64-bit unified addressing, caching, ECC memory, CUDA C, C++, OpenCL, and DirectCompute

Figura 1. Evolução das GPUs da NVIDIA [Nickolls and Dally 2010].

2. GPU e processamento paralelo

A partir de 2003, foram estabelecidas duas trajetórias principais para a produção de microprocessadores: *multicore* e *many-core*. Um processador *multicore* tenta manter a velocidade de execução de programas sequenciais enquanto trabalha em vários núcleos. Eles começaram com dois núcleos, e esse número tem dobrado a cada geração. Um exemplo é o microprocessador Intel Core i7, que tem oito núcleos de processamento, cada um dos quais é um processador de múltiplas instruções que implementa o conjunto de instruções x86 completo. Em contraste, um microprocessador *many-core* concentra-se mais no *throughput* de execução de aplicações paralelas. Eles começaram com um grande número de núcleos muito pequenos, com esse número também duplicando-se a cada geração. Um exemplo é a GPU NVIDIA GeForce GTX 280 com 240 núcleos, cada um dos quais é um processador de instrução única, em ordem e que possui múltiplas *threads*, o qual partilha a sua cache de instrução e de controle com sete outros núcleos [Kirk and Hwu 2010].

Desde então, as GPUs lideram a disputa de desempenho no processamento com ponto flutuante, como mostra a Figura 2. Enquanto a melhoria de desempenho das CPUs diminuiu significativamente, as GPUs continuaram a melhorar. Desde 2009, a relação entre GPUs e CPUs para o *throughput* do processamento com ponto flutuante é cerca de 10 para 1 (1 teraflops (1000 gigaflops) contra 100 gigaflops) com relação à velocidade que os recursos de execução podem suportar nestes chips. [Kirk and Hwu 2010].

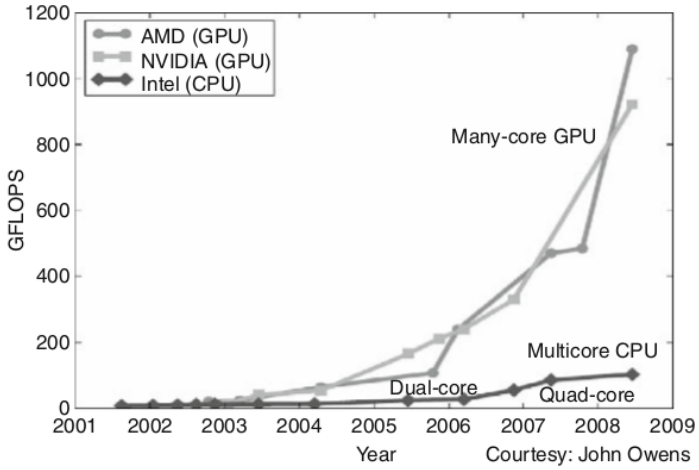

Figura 2. Diferença de desempenho entre GPUs e CPUs [Kirk and Hwu 2010].

Muitos desenvolvedores tem optado por executar as partes de computação intensiva de seus *softwares* em GPUs devido a grande diferença de desempenho entre a execução paralela e sequencial. Essa diferença entre o desempenho de GPUs e CPUs existe devido à diferença entre as filosofias de projeto, como mostra a Figura 3.

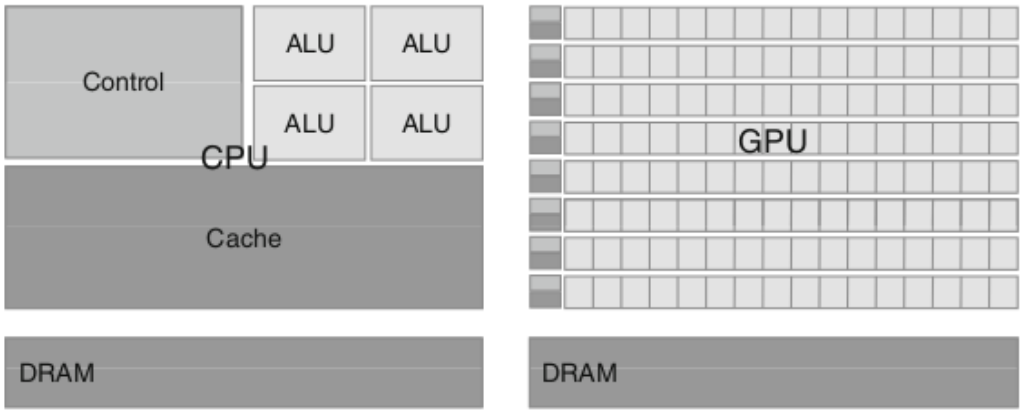

Figura 3. Comparação entre os projeto de uma CPU com quatro núcleos e de uma GPU [Kirk and Hwu 2010].

As CPUs foram projetadas com o objetivo de possuir alto desempenho em códigos sequenciais. No entanto, usa uma lógica de controle que permite a execução de instruções de uma única *thread* em paralelo ou mesmo fora de sua ordem sequencial mantendo a aparência de execução sequencial. Além disso, grandes memórias caches tentam reduzir as latências de acesso a instruções e a dados. Apesar disso, a execução de cálculos de forma rápida não é garantida. Desde 2009, o aumento no número de núcleos nas CPUs tem sido explorado a fim de melhorar o desempenho de códigos sequenciais [Kirk and Hwu 2010].

A filosofia de projeto das GPUs foi motivada pela necessidade de executar um grande número de cálculos de ponto flutuante por frame de vídeo em jogos avançados.

Isso fez com que os fornecedores de GPU optassem por um projeto que otimizasse o *throughput* de execução de um grande número de *threads*, a fim de maximar a área do chip e o gasto de energia dedicada a cálculos de ponto flutuante.

Uma característica do hardware de um GPU é que devido as muitas *threads*, enquanto algumas esperam por um acesso a memória que tem alta latência, as demais *threads* são aproveitadas para procurar tarefas a serem executadas. Essa característica minimiza o lógica de controle necessária para cada *thread*. Além disso, as pequenas memórias cache são projetadas para ajudar no controle de largura de banda demandada pelas aplicações de múltiplas *threads* minimizando o acesso a DRAM [Kirk and Hwu 2010].

É fácil perceber que as GPUs são projetadas para a execução de computação numérica e que as CPUs podem ter um desempenho melhor em algumas tarefas para as quais as GPUs não foram projetadas. Logo, a maioria das aplicações irão usar CPU, para as partes sequenciais e GPU para as partes de computação numérica intensa. Entretanto, alguns modelos de programação como o CUDA (*Compute Unified Device Architecture*), são projetados para suportar a execução de uma aplicação de forma conjunta na CPU e na GPU.

Outro fatores, além do desempenho, devem ser considerados pelo desenvolvedor na escolha de um processador. Considerando a necessidade de execução de computações numéricas, um desses fatores é o suporte para o padrão de ponto flutuante do IEEE (*Institute of Electrical and Electronics Engineers*). Desde a introdução do GeForce GTX 8800 ou simplesmente G80, em 2006, as GPUs oferecem suporte ao padrão do IEEE, apesar disso não ter sido comum nas primeiras gerações. Processadores de fornecedores distintos com suporte a este padrão fornecem resultados previsíveis. Além disso, as GPUs mais recente, possuem abordagens mais rápidas para execução de ponto flutuante de precisão dupla.

Outro fator, é popularidade do processador, principalmente, devido ao custo de desenvolvimento de uma aplicação sobre um sistema de computação paralela. Entretanto, com a popularidade da GPU no mercado de PC, a computação massivamente paralela foi caracterizada como um produto popular, o que tornou as GPUs economicamente atraentes para os desenvolvedores de aplicativos.

Quando é possível desenvolver uma aplicação adequada para execução paralela, uma implementação em GPU pode alcançar uma velocidade aproximadamente 100 vezes maior que uma execução sequencial. Existem diversos níveis de detalhamento do paralelismo que podem ser alcançado com a arquitetura CUDA que possui um modelo de programação para facilitar a implementação paralela e o gerenciamento do envio de dados [Kirk and Hwu 2010].

Além das inúmeras aplicações de computação atuais, muitas aplicações interessantes do futuro podem usufruir dos benefícios do alto desempenho da computação paralela como simulações detalhadas na área médica e biológica; codificação e manipulação de vídeo e áudio, síntese de imagens e exibição de alta resolução em televisões de alta definição (HDTV); manipulação de interfaces de usuário em telas sensível ao toque de aparelhos celulares e monitores de vídeo; e principalmente, aplicações no mercado de jogos, grandes investidores no desenvolvimento de GPUs.

3. GPGPU

Com a evolução dos projetos de harware de GPU na direção de processadores mais unificados, estes estão cada vez mais semelhantes aos computadores paralelos de alta performance. Alguns pesquisadores começaram a perceber o avanço de desempenho das GPUs e começaram a explorar o seu uso para resolver problemas de cálculos intensivos de engenharia e ciência. Entretanto, as GPUs haviam sido projetadas apenas para características requerida por APIs gráficas. Para acessar recursos computacionais, um programador tinha que utilizar operações gráficas nativas através de funções OpenGL ou DirectX [Kirk and Hwu 2010].

O início da utilização das GPUs para programação de aplicações de propósitos gerais impulsionou o desenvolvimento de facilidades na arquitetura das GPUs e principalmente, na programação com a introdução da arquitetura CUDA (*Compute Unified Device Architecture*) apresentada pela NVIDIA, em 2006 [NVIDIA b]. A CUDA tem o objetivo de facilitar a interface entre o programador e as aplicações GPU. Com isso, surgiu o termo GPGPU (*General-Purpose Computing on Graphics Processing Units*).

A arquitetura CUDA é um modelo de programação e uma plataforma de computação paralela que pretende melhorar o aproveitamento do poder de processamento da GPU. O modelo de programação CUDA fornece aos programadores abstrações simples da organização hierárquica das *threads*, sincronização e memória permitindo a implementação adequada de programas para a GPU. A CUDA oferece suporte a linguages de programação como C, C++, Fortran, OpenCL e DirectX [NVIDIA b].

A GPU G80 da NVIDIA foi a primeira geração de GPUs preparada para CUDA e suporte a linguagem C, o que tem melhorado cada vez mais na gerações seguintes: G200, lançada em 2008; Fermi, lançada em 2009 e; Kepler, lançada em 2012.

4. Arquitetura de uma GPU

As GPUs da NVIDIA com suporte a CUDA são organizadas em um vetor de SMs (*Streaming Multiprocessors*) com múltiplas *threads*. Na Figura 4, dois SMs formam um bloco de construção (o número de SMs pode variar de uma geração para outra geração). Cada SM tem um número de SPs (*Streaming Processors*) que compartilham a lógica de controle e a cache de instruções. Cada GPU vem com até 4 GB de DRAM com taxa de dados duplos gráficos (GDDR). Essas DRAMs GDDR diferem das DRAMs de sistema na placamãe da CPU pois são essencialmente a memória usada para gráficos. Para aplicações gráficas, elas envolvem imagens de vídeo e informação de textura para renderização 3D, mas para computação elas funcionam como uma memória fora do chip com largura de banda muito alta, porém com um pouco mais de latência do que o sistema de memória típico. A arquitetura CUDA foi introduzida pela GPU G80 que tem 86.4 GB/s de largura de banda de memória, mais 8 GB/s de largura de banda de comunicação com a CPU, com esta devendo crescer conforme a largura de banda do barramento da CPU da memória do sistema cresça [Kirk and Hwu 2010].

A GPU G80 é massivamente paralela e internamente seu chip possui 128 SPs (16 SMs, cada um com 8 SPs), totalizando 500 gigaflops. Cada SP tem uma unidade de multiplicação-adição (MAD) e uma unidade de multiplicação. Outras unidades de função especial executam funções de ponto flutuante tais como raiz quadrada (SQRT). Cada SP

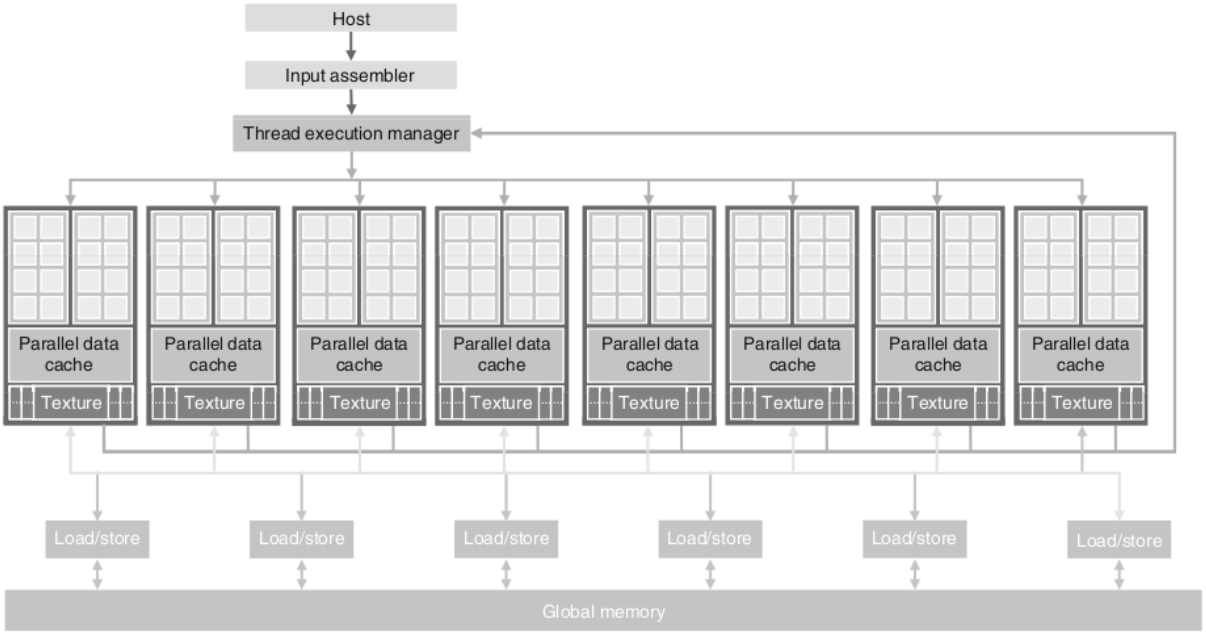

Figura 4. Arquitetura de uma GPU com suporte a CUDA [Kirk and Hwu 2010].

pode rodar milhares de *threads* por aplicação. Uma boa aplicação roda tipicamente entre 5000 à 12000 *threads* simultaneamente. O chip G80 suporta até 768 *threads* por SM, que soma cerca de 12000 *threads* para este chip. Com 240 SPs, o GT200, mais recente, ultrapassa 1 teraflops e suporta 1024 *threads* por SM e até cerca de 30.000 *threads* para o chip. Comparando com *multithreading* simultâneo, uma CPU Intel típica suporta em média 2 ou 4 *threads*, por núcleo [Kirk and Hwu 2010].

4.1. Arquitetura da GPU Fermi

A arquitetura Fermi, lançada em abril de 2010, é apresentada nas Figuras 5 e 7. Esta arquitetura trouxe suporte para novas instruções para programas em C++, como alocação dinâmica de objeto e tratamento de exceções. Cada SM possui 32 núcleos CUDA, um total de 512 núcleos. Até 16 operações de precisão dupla por SM podem ser executadas em cada ciclo de clock. Além disso, cada SM possui 16 unidades de *load* e *store*, possibilitando que o endereço de fonte de destino possam ser calculados para 16 *threads* por clock; 4 SFUs (*Special Function Units*), que executam instruções transcendentais, como seno, cosseno, raiz quadrada, etc. Cada SFU é desacoplada da unidade de emissão, permitindo que esta possa realizar a sua emissão de outra instrução enquanto a SFU está ocupada [NVIDIA a].

A arquitetura GPU balança seu poder de computação paralela com controladores de memória DRAM paralela projetados para largura de banda de memória alta. A GPU Fermi tem 6 interfaces DRAM GDDR5 de alta velocidade, cada uma com 64 bits de largura. Seus 40 bits de endereço lidam com até 1 Tbyte de espaço de endereçamnte para DRAM GPU [Kirk and Hwu 2010].

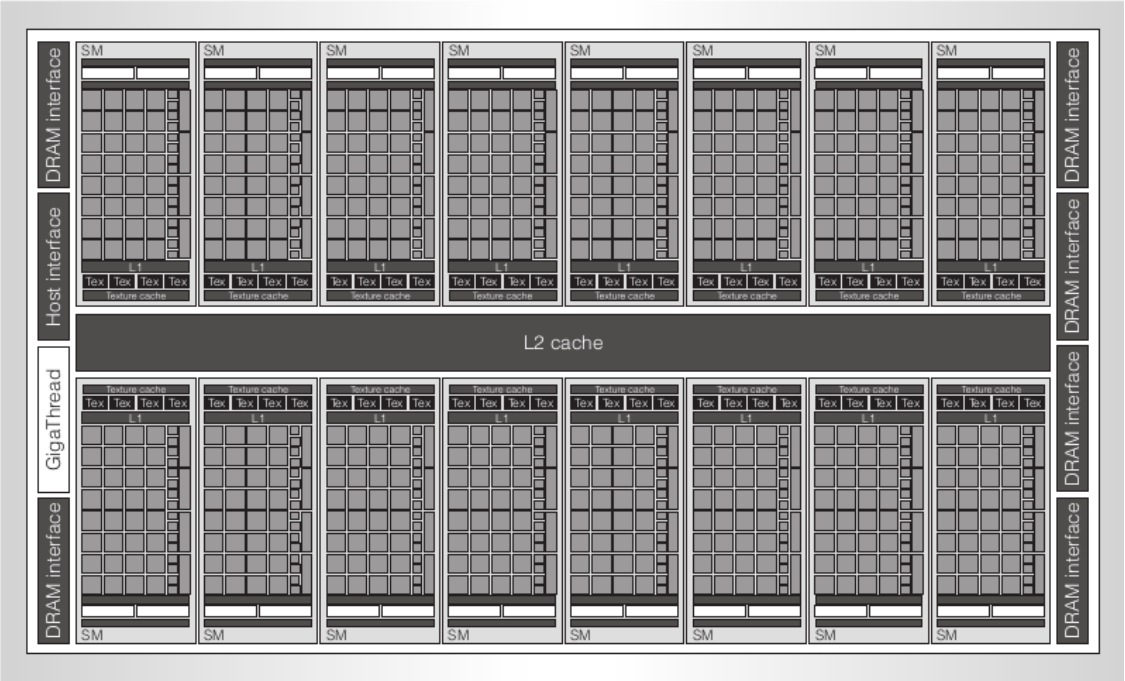

Figura 5. Arquitetura da GPU Fermi [Nickolls and Dally 2010].

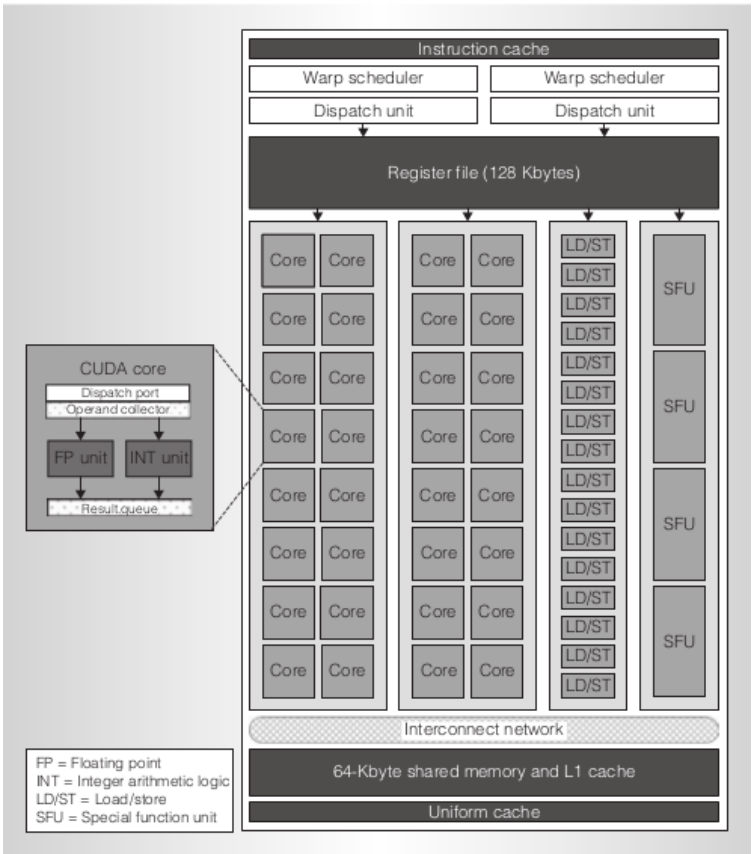

Figura 6. Arquitetura de um microprocessador da GPU Fermi [Nickolls and Dally 2010].

5. Arquitetura Unificada

Muitos fabricantes de processadores estão optando pelo arquitetura unificada, isto é, CPUs e GPUs dentro de um único chip. Apesar de aumentar o desempenho dos processadores e melhorar a eficiência energética, o núcleo da CPU e o núcleo da GPU continuam trabalhando quase exclusivamente em funções separadas. Eles raramente colaboram para executar um determinado programa, não sendo tão eficientes quanto poderiam ser.

As grandes empresas concorrentes no mercado de microprocessadores são a NVIDIA, a Intel e a AMD. Em 2006 houve uma fusão entre a ATI e AMD, a qual neste mesmo ano anunciou o ínicio do desenvolvimento de uma tecnologia integrando GPU e CPU, oficializando o anúncio em 2010 com o lançamento do processador *AMD Fusion*. Logo em seguida, em 2011, a Intel divulgou a nova microarquitetura *Sandy Bridge*, com o mesmo propósito de integração com GPU da NVIDIA.

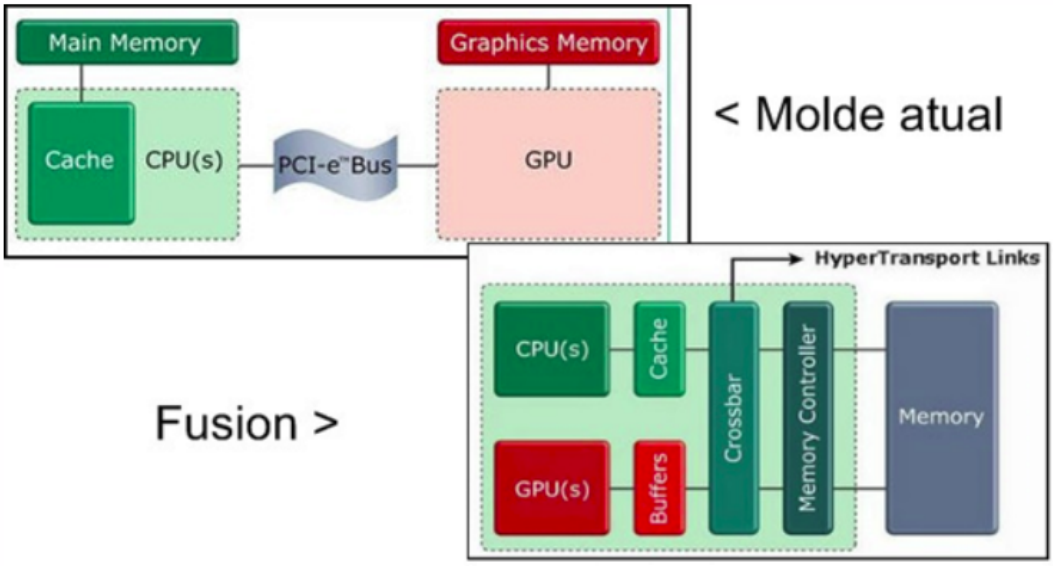

Figura 7. Organização interna do AMD Fusion [Augusto].

Uma nova técnica para arquitetura unificada é apresentada por Yang et al. [Yang et al. 2012]. Segundo testes apresentados por Yang et al., esta técnica melhora o desempenho de computadores em aproximadamente 20% em comparação com os mesmos núcleos trabalhando em funções separadas.

A interligação entre processadores gráficos e processadores principais não exige nenhuma tecnologia nova de hardware e consiste em fazer com que as GPUs colaborem com as CPUs, permitindo que os núcleos GPU executem as funções de cálculo, enquanto as CPUs carregam e preparam os dados vindos da memória que as GPUS vão precisar. Esta técnica tira proveito da rapidez das GPUs na execução de cálculos individuais, e da flexibilidade das CPUs em desempenhar tarefas mais complexas. Como as GPUs processam os dados mais rapidamente, as CPUs ficam encarregadas de descobrir o que a GPU irá precisar a seguir, busca esses dados na memória e os entrega prontos. Isso elimina a necessidade da GPU consultar a memória e consequentemente seu desempenho melhora já que concentra-se no processamento dos dados [Yang et al. 2012].

6. Utilização em Sistemas de Paralelismo Massivo (MPP)

Com desenvolvimento de facilidades na arquitetura das GPUs e principalmente, na programação, as GPUs começaram a ser bastante utilizadas também em sistemas de paralelismo massivo. Um exemplo é o supercomputador Titan. Lançado em 2012, é considerado pelo Top500 o mais rápido do mundo, com uma capacidade de processamento de 20 petaflops [Top500].

O Titan é um sistema de paralelismo massivo Cray XK7, com 18.688 nós de computação em 200 gabinetes. Cada nó é potencializado com um processador de 2.2 GHz AMD 16-core Opteron 6274 com 32GB de memória DDR3 e uma GPU NVIDIA K20 Tesla com 6GB de memória de alta velocidade. Ao todo, a máquina tem 710 TB de memória. Devido a eficiência no uso de energia e do custo benefício da GPU Tesla K20, o Titan é 10 vezes mais rápido e cinco vezes mais eficiente do ponto de vista energético do que seu predecessor, o sistema Jaguar de 2,3 petaflops, sem ocupar mais espaço [NVIDIA c].

7. Conclusão

O número de núcleos vêm crescendo em proporção ao crescimento da disponibilidade de área do chip do processador. Em paralelo, a arquitetura das GPUs continua evoluindo e cada vez mais proporcinando alto desempenho no processamento de gráficos e cálculos numéricos, tendendo a se generalizar com o desenvolvimento de novas tecnologias GPGPU.

Apesar de seu alto poder de computação em aplicações paralelas, o projeto de processadores de núcleo GPU é simples, o que vêm mudando nos últimos anos com a introdução de técnicas cada vez mais sofisticadas a fim de aumentar a utilização das unidades de cálculos.

Muitas vantagens da GPU ainda não são totalmente aproveitadas devidas as dificuldades que ainda existem na programação de *softwares*, o que também tende a melhorar com a desenvolvimento de novas técnicas, linguagens, compiladores e aprimoramento dos modelos de programação.

Devido a computação paralela escalável em GPU ser um campo recente, muitas aplicações estão sendo criadas rapidamente, o que impulsiona as empresas desenvolvedoras de processadores a descobrirem e implementarem novas otimizações cada vez mais rápido. E consequentemente, o número de sistemas de paralelismo massivo que utilizam GPU tendem a crescer, juntamente com o crescimento do desempenho desses sistemas.

Referências

Augusto, D. Amd fusion. Disponível em: `http://www.iotecnologia.com.br/amd-fusion`. Acessado em: 13 Nov 2012.

Kirk, D. B. and Hwu, W.-m. W. (2010). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Nickolls, J. and Dally, W. J. (2010). The GPU Computing Era. *IEEE Micro*, 30(2):56–69.

NVIDIA. Fermi computer architecture whitepaper. Disponível em: http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf. Acessado em: 13 Nov 2012.

NVIDIA. Introducing CUDA 5. Disponível em: http://www.nvidia.com/object/cuda_home_new.html. Acessado em: 13 Nov 2012.

NVIDIA. Supercomputador Titan opera com mais de 18 mil GPUs NVIDIA. Disponível em: http://www.nvidia.com.br/object/prbr_103012.html. Acessado em: 13 Nov 2012.

Top500. Oak Ridge Claims No. 1 Position on Latest TOP500 List with Titan. Disponível em: http://www.top500.org. Acessado em: 13 Nov 2012.

Yang, Y., Xiang, P., Mantor, M., and Zhou, H. (2012). Cpu-assisted GPGPU on fused CPU-GPU architectures. In *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, HPCA '12, pages 1–12, Washington, DC, USA. IEEE Computer Society.