# Graphics Processing Unit (GPU)

**Gabriel Matheus Oliveira**[1]

[1]Universidade Federal de Viçosa (UFV)
Rio Paranaíba – MG – Brasil

***Abstract.*** *This paper describes the activities and operation of Graphics Processing Units (GPUs), from their use in image processing to their applications in enterprise environments. It also discusses specific hardware and software architectures designed to support this type of processor, as well as the continuous evolution of GPU technologies.*

***Resumo.*** *Este trabalho descreve atividades e funcionamento de GPU's, desde sua utilização em processamentos de imagens, como em suas aplicações empresariais. Fala também de uma arquitetura de hardware e software específicos para tratarem desse processador e sua situação de constante evolução.*

## 1. Introdução

Antes da chegada e do surgimento das poderosas *GPUs*, quem realizava as tarefas que hoje são das *GPUs*, era a *CPU*, através de controladores *VGA*. No entanto, compreendeu-se que isso não era o bastante para a demanda exigida de cálculos específicos do mercado [Amorim and da Silva Araújo 2011]. Mediante este cenário, no fim da década de 1990, a *NVIDIA* lançou a primeira placa de vídeo denominada *GeForce 256*. No contexto da época, o objetivo principal da *GPU*, era aumentar a rapidez de imagens, vídeos e gerações de gráficos *3D* e *2D*, que posteriormente seriam utilizados por sistemas operacionais [Zanotto et al. 2012].

Grande parte do desenvolvimento da *GPU*, ocorre em razão da plataforma *Compute Unified Device Architecture - (CUDA)* [Zanotto et al. 2012, Cook 2013]. Ela se trata de uma arquitetura de *hardware* e *software* que proporciona um aumento significativo de desempenho de programas escritos em diversas linguagens de alto nível na *GPU*.

Atualmente, a Unidade Gráfica de Processamento tem sido essencial para a performance computacional, em processamento de imagens nos gráficos de vídeos e jogos. No presente trabalho, será descrito como se dá o funcionamento das *GPUs*, relacionando suas funções com as atividades de maior abrangência.

Este trabalho tem com objetivo descrever atividades e funcionamentos das *GPUs*, desde sua utilização em processamento de imagens, vídeos e aplicações empresariais. Além disso, são descritas as arquiteturas de *hardware* e *software* das *GPUs*.

O restante deste artigo é organizado da seguinte forma: a Seção 2 descreve as diferenças entre *CPU* e *GPU*. A Seção 3 descreve o modelo de programação *CUDA*, o qual é utilizado em *GPUs*. A Seção 4 aborda a paralelização em *GPU* e seus benefícios. E a Seção 5 fala da importância de *GPU* em aplicações em *Deep Learning* e na sequência são apresentadas as conclusões na Seção 6.

## 2. *CPU* x *GPU*

### 2.1. O que são *CPU* e *GPU*

A *CPU - (Central Processing Unit* ou Unidade Central de Processamento) é o processador da máquina, pode-se dizer que ela é o cérebro do aparelho, tem como função organizar as atividades e funções que o usuário requer do dispositivo. Quanto mais poderosa é a *CPU*, mais cálculos por segundo ela realiza, e por consequência mais rápida ela é. Para o usuário, essa velocidade é transmitida pela agilidade que programas são executados, especialmente os que fazem cálculos mais pesados como por exemplo programas de modelagem em *3D* e de manipulação de imagens [Stallings 2009].

A *GPU - (Graphic Processing Unit* ou Unidade de Processamento Gráfico) comumente chamada de placa de vídeo, é o componente dedicado a processar toda a parte gráfica exibida na tela. É especialista em cálculos de processamento de vetores, texturas, desenhos, entre outros, de maneira muito rápida, liberando a *GPU* para fazer outras tarefas. O uso de uma *GPU* ocasiona uma melhora significativa do que é exibido na tela, já que retira essa função de processamento da *GPU* que mesmo conseguindo fazer muitos cálculos em um curto espaço de tempo não é especialista neste tipo. Um *desktop* ou *notebook* equipado com *GPUs* poderosas e dedicadas podem rodar com muito mais fluidez e desenvoltura jogos e aplicações com aceleração gráfica habilitada, como por exemplo editores de vídeo. Quanto mais poderosa a *GPU* mais elementos podem ser mostrados na tela, com mais efeitos como luz, sombras e alto detalhamento [Anderson et al. 2008, Cook 2013].

### 2.2. Principais diferenças entre *CPU* e *GPU*

A principal diferença entre os dois componentes é que a *CPU* realiza qualquer tipo de cálculo de processamento, incluindo os gráficos, porém esses cálculos são realizados de forma mais lenta. Por esse motivo existem as *GPUs*, elas são responsáveis por calcular os processamentos de vetores, texturas e desenhos que compõem as imagens que aparecem na tela, tudo isto de forma rápida. Os processadores são mais lentos, porém são mais flexíveis e se adaptam a uma gama maior de tarefas e conseguem realizar tarefas complexas com maior facilidade do que uma placa de vídeo.

Toda vez que o usuário assiste um filme em tela cheia, abre uma imagem ou joga, seja por *videogame*, *smartphone* ou computador, a *GPU* está gerenciando cada *pixel* que aparece na tela ao realizar essas funções. Esses pontos estão em constante mutação e a *GPU* administra as variadas situações a que cada ponto está vinculado em termos de luminosidade e cor, tudo isso em instantes muito curtos.

### 2.3. Se a *GPU* é mais rápida, por que ainda se usa *CPUs*?

Os processadores gráficos presentes na geração atual de placas de vídeo são extremamente poderosos, e as fabricantes a cada lançamento vem dando cada vez mais destaque a toda essa potência. Mesmo que os processadores gráficos sejam mais potentes que as *CPUs*, eles não podem substituí-las pois não são processadores centrais, o processador instalado na placa mãe se conecta aos mais diversos componentes da máquina, incluindo as placas de vídeo.

Enquanto o processador tem as mais diversas tarefas, como de controlar todas as outras peças e o de gerenciar e abrir aplicativos, a *GPU* é responsável somente pela parte

de vídeo em qualquer aplicação, seja ela a interface gráfica do seu dispositivo, o vídeo que você assiste no *YouTube* ou qualquer outra parte.

Com isso, depreende-se que mesmo com uma maior potência, a *GPU* não executa cálculos de alta complexidade com a mesma desenvoltura que uma *CPU*, enquanto a *CPU* possui uma grande flexibilidade e pode realizar diversos cálculos distintos, a *GPU* é especializada em cálculos simples, por essa razão, uma não pode tomar o lugar da outra na construção da máquina. A *CPU* é focada em tarefas que demandam mais cálculos e operações complexas que necessitam de poucos núcleos e *clocks* mais altos. A *GPU* é mais voltada para cálculos simples em múltiplos núcleos, sendo que ela trabalha com *clocks* mais baixos [Stallings 2009, Cook 2013].

## 3. *Compute Unified Device Architecture - CUDA*

O *CUDA*, é uma arquitetura de *hardware* e *software* que produz um ambiente de programação que possibilite as *GPUs* executarem programas escritos em linguagens de programação de alto-nível, dentre essas linguagens, pode-se incluir a linguagem *C*, *Fortran*, *OpenCL*, entre outras [Zanotto et al. 2012].

O *CUDA kernel* (*kernel* é apenas o nome dado as funções que executam na *GPU*) executa através de um conjunto paralelo de *threads*, cada *thread* possui um *ID* utilizado para armazenar o endereço de memória e tomar decisões de controle. O compilador e/ou programador, organiza as *threads* em blocos; posteriormente em uma grade de blocos, a *GPU* chama o *kernel* em uma grade de blocos de *threads*. Particularmente, as *threads* dentro dos blocos executam a chamada do *kernel*. Um bloco de *threads* é um conjunto de *threads* que trabalham ao mesmo tempo, podendo cooperar entre si, utilizando-se da *barrier synchronization* e da memória compartilhada, cada bloco de thread também possui um ID dentro da grade. Uma grade é conjunto ordenado de blocos que executa o mesmo *kernel*, faz leituras de entradas e atribuições de saída na memória global [Cook 2013], [Kirk et al. 2007].

No modelo de programação *CUDA*, cada *thread* tem uma memória privada, cada bloco de *thread* tem uma memória compartilhada pelo bloco, essa memória é usada para fazer comunicação entre as *threads*, transferir dados e para compartilhar os resultados das operações realizadas. Uma grade compartilha os resultados das operações com a memória global após a sincronização global do *kernel* [Cook 2013].

## 4. Paralelização em *GPU*

Como visto, a *GPU* é uma unidade de processamento especializado em renderização de gráficos 3D. Uma característica marcante de uma *GPU* é a capacidade de processar trechos de código em paralelo de maneira muito eficiente, sua principal diferença para uma *CPU* é que, enquanto as *CPUs* dedicam grande parte de seus circuitos ao controle, uma *GPU* foca mais nas unidades aritméticas.

Em primeiro lugar, deve-se lembrar que esse poder de processamento paralelo, torna a execução de algoritmos complexos mais eficiente. Isso sugere que grandes problemas podem muitas vezes ser divididos em vários problemas menores, que podem ser resolvidos ao mesmo tempo mais rapidamente.

Outro aspecto a se destacar é a evolução contínua das *GPUs*, isso se dá pela

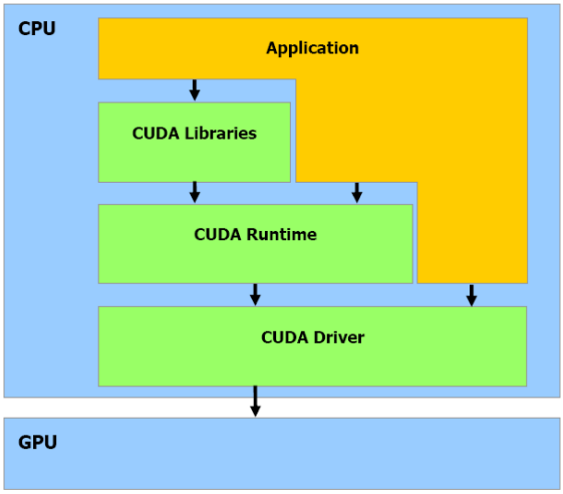


**Figura 1. Transformação CPU/GPU por CUDA**

necessidade de problemas mais complexos, como visualização de gráficos, processamento de grande quantidade de cálculos numéricos e flexibilidade de programação. Com a constante evolução, provocou-se uma discussão entre especialistas como diz [Rodrigues 2013], e incentiva-se cada vez mais a completa substituição das *CPUs* tradicionais por *GPGPUs*, ou seja, unidades de processamento gráfico de propósito geral [Cook 2013].

### 4.1. Benefícios do paralelismo

O paralelismo proporciona muitos benefícios, entre eles a aplicação no campo da física, matemática, estudos fármacos, estruturas biológicas e várias outras áreas do campo científico. Por sua alta capacidade em vetores e matrizes, também se consegue utilizar em empresas, como por exemplo em:

- Consultas de banco de dados
- Busca de força bruta em criptografia
- Simulações de computador comparando muitos cenários independentes
- Geo-visualização
- Inteligência Artificial

## 5. *GPU* e *Deep Learning*

### 5.1. Importância

Ao decorrer dos anos a importância das *GPUs* tem crescido muito, e no cenário atual, tem se mostrado uma boa opção para serem usados em *Machine Learning* (aprendizado de máquina) na resolução dos mais variados problemas, não apenas sendo usados para fazerem análises de dados em empresas. A empresa que mais tem se destacado nessa área é a *NVIDIA*, que buscou desenvolver uma tecnologia voltada a resolver os problemas matemáticos mais intensivos que demandam maior poder de processamento. Tudo isso foi possível através da criação da linguagem de programação *CUDA - (Compute Unified Device Architecture)*, que permite que as suas *GPUs* processem uma imensa quantidade de dados. Presente em *mainframes*, e até mesmo em sistemas mais modestos voltados

para usuários domésticos, essa é a forma que a *NVIDIA* tem fornecido para que as *GPUs* ajudem esses problemas a serem sanados.

As *CPUs* - (Unidades de Processamento Central) têm recebido cada vez mais ajuda das *GPUs*, que fornecem um poder extra e processam as tarefas de maior complexidade. Exemplo: jogos pesados, vídeos e imagens em alta definição e assim por diante. Tudo isso é possível, pois a arquitetura das *GPUs* é feita para executar uma tarefa seguidas vezes, de forma rápida, sendo bastante útil para um processamento iterativo, o que possibilita serem usadas para algoritmos de aprendizado de máquina.

### 5.2. Diferenças entre *CPU* X *GPU*

Para se ter uma ideia da diferença de ambas, as *CPUs* possuem poucos núcleos que executam uma determinada tarefa, e não são bem otimizadas para rodarem tarefas que exigem um alto poder de processamento. Já a *GPU* foi criada para suportar uma carga muito maior, possuindo milhares de núcleos que são capazes de tratar uma grande quantidade de tarefas de forma simultânea.

### 5.3. Computação acelerada pela *GPU*

A computação acelerada pela *GPU* funciona usando as *GPUs* em conjunto com as *CPUs* para acelerar a análise de dados, programas e ou outras aplicações com fins analíticos. Esse processo transfere o processamento mais complexo para a *GPU*, que consegue resolver as tarefas mais exigentes. A *GPU* recebe parte da tarefa e a *CPU* recebe uma outra parte. Usando o método da paralelização, tornando a execução dessas tarefas mais rápida. Cada vez mais empresas como a *Microsoft* utilizam sua tecnologia *(Direct Compute)*, é só uma questão de tempo para o que o processamento em *GPU* seja uma realidade adotada pelo mercado.

### 5.4. Uso em aplicações analíticas

Empresas como *Google*, *Microsoft*, *Facebook*, têm optado por usar as *GPUs* para *Deep Learning* por fornecerem um tempo menor para treinaremto de algoritmos, pois tarefas que demandam dias podem ser executadas em questão de horas, resultando em impactos de maior significância na obtenção de ótimos resultados, principalmente em campos de visão computacional, reconhecimento de voz e processamento de imagens utilizando esses algoritmos em conjunto com as *GPUs*.

### 5.5. *Deep Learning* e sua importância no cenário atual

A *Deep Learning* tem crescido e se destacado na área de inteligência artificial, ajudando as máquinas a compreender dados, imagens, sons e textos. Usando redes neurais, as máquinas são capazes de compreender e distinguir cada vez mais tarefas complexas, que antes somente os humanos podiam fazer com êxito [Goodfellow et al. 2016]. Uma área que com certeza contribuirá para seu crescimento e desenvolvimento, é a *IOT* - (Internet das Coisas), fazendo com que máquinas ou objetos sejam capazes de entender o que os seres humanos desejam [Gubbi et al. 2013].

Assim como foi citado anteriormente, a *NVIDIA* além de ser a empresa que mais se destaca nesse ramo, tem treinado excessivamente as *GPUs*, para detectarem imagens e fazerem a diferenciação entre os objetos. Cada vez mais, as máquinas têm se mostrado

não somente compreensivas, mas também capazes de tomar decisões mais expressivas, e o mais espantoso é que em apenas 3 anos as placas de vídeo aceleraram o treinamento e o desenvolvimento das redes neurais de 10 a 20 vezes, em um ritmo mais rápido que a Lei de *Moore* aponta. Toda essa evolução pode vir a ser aplicada em várias áreas do conhecimento, como medicina, engenharia e automação. Existe o caso das empresas que rapidamente perceberam isso e cada vez mais buscam usar essa tecnologia para lidar, interpretar e ajudar na tomada de decisões dentro de cada uma delas, agregando valor em empresa por empresa, tornando possível melhorar seus serviços e produtos, e por último e não menos importante proporcionando uma vantagem competitiva e inovadora [Brown 2015].

Modelos *Deep Learning* requerem a utilização de um *framework* a fim de facilitar o processamento de redes neurais com bibliotecas pré-construídas e habilitadas para processamento em paralelo. A Tabela 1 lista os principais *frameworks* para *Deep Learning* e apresenta informações sobre as linguagens e sistemas operacionais suportados, bem como o modelo de programação de *GPU* e bibliotecas relacionadas, as quais podem ser utilizadas para facilitar a programação e utilizar o *framework* escolhido como *backend* [NVIDIA 2018].

**Tabela 1. Principais *frameworks* para *Deep Learning***

| Framework | Linguagem | Sistema Operacional | GPU | Bibliotecas Relacionadas |
|---|---|---|---|---|
| Caffe | C++<br>Python<br>Matlab | Linux<br>Windows<br>Mac | CUDA<br>Opencl | Lasagne<br>Keras |
| Tensorflow | Python | Linux<br>Windows<br>Mac | CUDA | Keras<br>Skflow |
| Theano | Python | Linux<br>Windows<br>Mac | CUDA<br>Opencl | |
| Torch | Lua<br>C | Linux<br>IOS<br>Android | CUDA | |
| MXNet | Python<br>R<br>Julia | Linux<br>Windows<br>Mac | CUDA | |

## 6. Conclusão

A *GPU*, portanto, proporciona um alto desempenho no processamento de gráficos e cálculos numéricos através de suas divisões em paralelo. Muitas vantagens da *GPU* ainda

não são totalmente aproveitadas devido às dificuldades que ainda existem na programação de *softwares*, o que pode melhorar com o desenvolvimento de novas técnicas, linguagens, compiladores, entre outros. O número de sistemas de paralelismo massivo aumentou, supercomputadores vêm se sujeitando cada vez mais à essa característica, fazendo com que melhorem gradativamente seu processamento. Além disso, a migração para *GPU*, está também impulsionando as empresas que desenvolvem processadores, para implementar e descobrir novas otimizações e ter mais competitividade no mercado, conseguindo atender esse nível de desempenho.

## Referências


Amorim, P. H. J. and da Silva Araújo, F. R. (2011). GPU: A matriz de processadores. *Instituto de Computação - UNICAMP*.

Anderson, J. A., Lorenz, C. D., and Travesset, A. (2008). General purpose molecular dynamics simulations fully implemented on graphics processing units. *Journal of Computational Physics*, 227(10):5342 – 5359.

Brown, L. (2015). Deep learning with GPUs. *Larry Brown Ph. D., Johns Hopkins University*.

Cook, S. (2013). *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

Kirk, D. et al. (2007). NVIDIA CUDA software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104.

NVIDIA (2018). Deep learning frameworks. Disponível em: `https://developer.nvidia.com/deep-learning-frameworks`. Acesso em: 28 de junho de 2018.

Rodrigues, E. d. C. S. (2013). GPU: sob o ponto de vista de arquiteturas paralelas, organização interna e utilização em sistemas de paralelismo massivo. *Instituto de Computação - UNICAMP*.

Stallings, W. (2009). *Computer Organization and Architecture: Designing for Performance*. Prentice Hall Press, Upper Saddle River, NJ, USA, 8th edition.

Zanotto, L., Ferreira, A., and Matsumoto, M. (2012). Arquitetura e programação de GPU Nvidia. *Instituto de Computação - UNICAMP*.