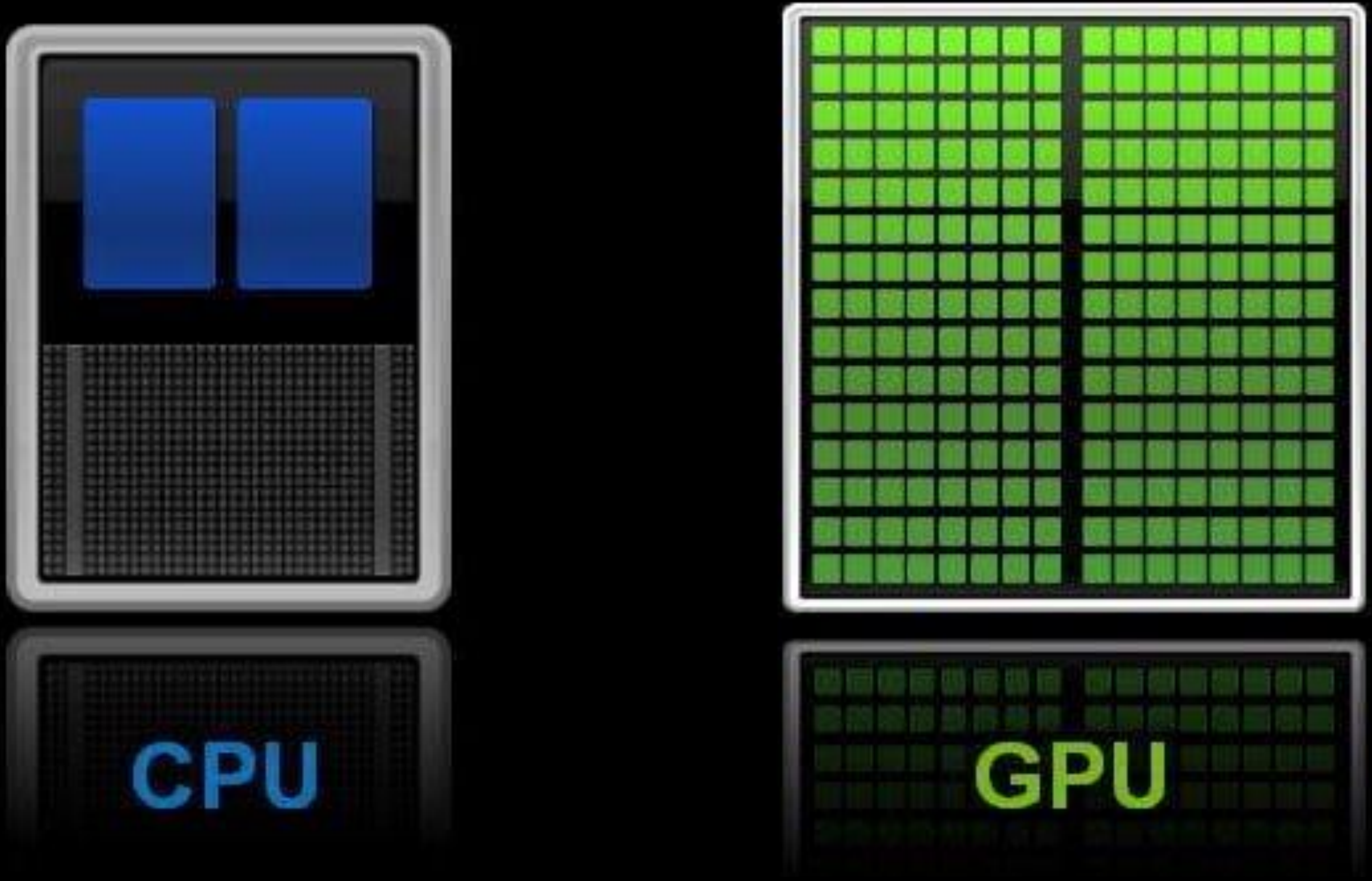
A GPU primer: Tips for getting started with GPUs for research applications

Domaniç Lavery and Siddharth Varughese

Infinera Inc.

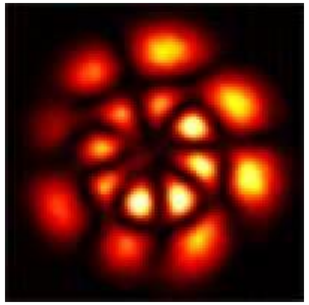
The Difference between a CPU and GPU

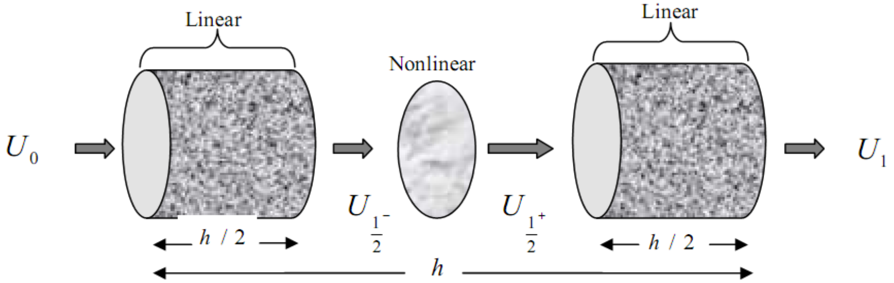


https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

Why should you use GPUs for research?

- Software-configurable transceiver modelling [1]
- ‘Hologram’ generation for **SLM mask** (multimode fibers [2])
- Fiber propagation simulations, e.g., **split step Fourier method**
- More generally, visualisations, graphics, anything involving an FFT or parallel processing





[1] NTT Digital Object Identifier 10.1109/ACCESS.2019.2904083
[2] Joel Carpenter, University of Queensland, probably several others

Why should you use ***not*** GPUs for research?

- When you need to do **one task** (not parallel)
- When you haven’t tried to optimize your *CPU* code
- When you need to **prototype code** quickly
- When you need to **easily debug** your code*

* Some exceptions, but it’s always more painful

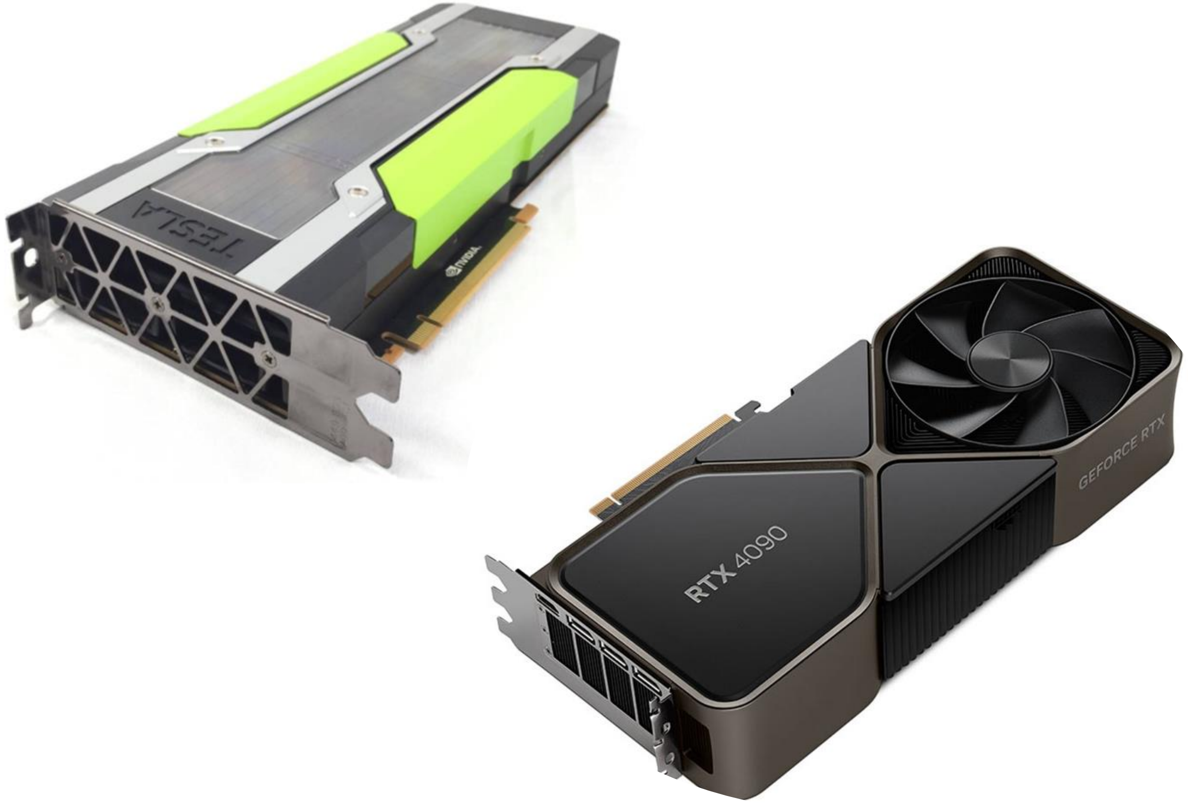
The easiest way...

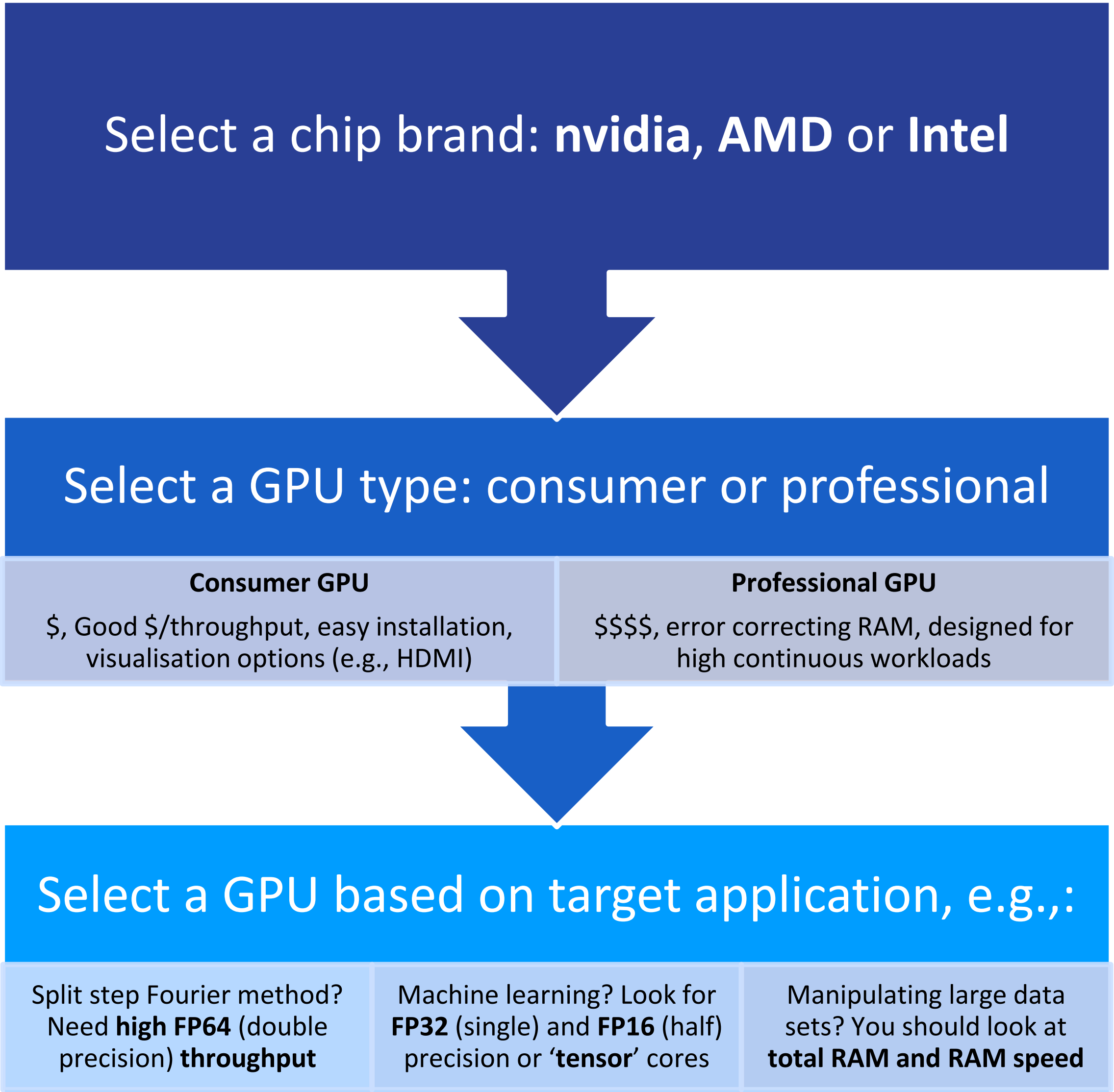
1. Buy a computer with any (recent) ***nvidia*** GPU
2. Install the Matlab ***Parallel Computing Toolbox****
3. Store data as **GPU arrays**, and most functions will execute on the GPU
4. At the end, run the '**gather**' function to return data to the CPU domain

- Not much else to say about this…

*https://uk.mathworks.com/products/parallel-computing.html
**https://uk.mathworks.com/help/parallel-computing/gpuarray.html

Common pitfalls and tips for purchasing and set up





Select a chip brand: **nvidia, AMD** or **Intel**

Select a GPU type: consumer or professional

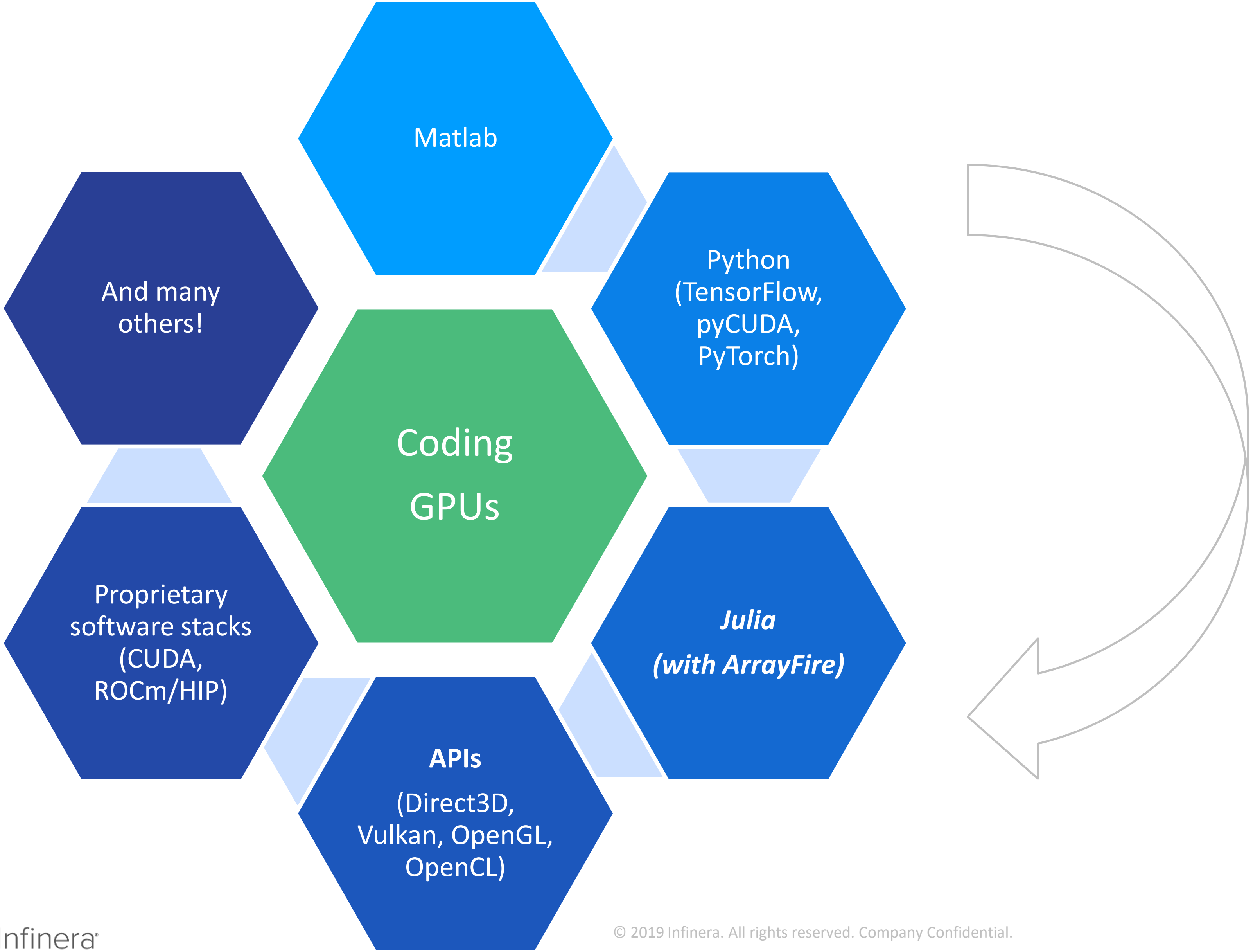
Consumer GPU	**Professional GPU**
$, Good $/throughput, easy installation, visualisation options (e.g., HDMI)	$$$$, error correcting RAM, designed for high continuous workloads

Select a GPU based on target application, e.g.,:

Split step Fourier method? Need **high FP64** (double precision) **throughput**	Machine learning? Look for **FP32** (single) and **FP16** (half) precision or ‘**tensor**’ cores	Manipulating large data sets? You should look at **total RAM and RAM speed**

techpowerup.com a great resource for comparing hardware!

Coding for a GPU


Matlab
Python (TensorFlow, pyCUDA, PyTorch)
Julia
(with ArrayFire)
APIs
(Direct3D, Vulkan, OpenGL, OpenCL)
Proprietary software stacks (CUDA, ROCm/HIP)
And many others!
Coding
GPUs

End user complexity increases clockwise

Julia demo and discussion

- This demo using Julia (language): https://julialang.org/
- With ArrayFire (API): https://arrayfire.com/
  - Julia ArrayFire bindings: https://github.com/JuliaGPU/ArrayFire.jl
  - And OpenCL under-the-hood (you shouldn’t need to worry about this one, it comes with your graphics driver)
- All are free and straightforward to install
  - any issues are usually PATH or Environment Variables: an Internet search will help

Considerations for installing GPUs

Internal

- Is the GPU physically too big for my machine? (single-, dual- or triple slot?)
  - usually an issue with upgrading prebuilt machines, e.g., Dell, Lenovo, HP...
- Is the GPU **compatible** with my **motherboard**?
- Is my **power supply** capable enough?
- Will it ***catch fire?***

eGPU

- Uses an enclosure to house to the GPU (see demo)
- Requires a high speed interface on host machine:
  - **Thunderbolt** (commonly available on laptops)
  - **OCulink** (not widely available, but looks great)
- If *data stays on the GPU*, interface speed is not a problem!
  - (Note: not true for games or visualisations)

Cloud

- e.g. Amazon EC2 (others are available)
- Expensive, but scalable
- Prototype on your own machine, then use for one-time huge simulations

Advanced Discussion Points

- CPU optimisations
  - Intel OneAPI
  - AVX

- Shader programming
  - Getting the most from a GPU