Moving To Vulkan

Asynchronous Compute

Chris Hebert, Dev Tech Software Engineer, Professional Visualization

Who am I?

Chris Hebert

@chrisjhebert

Dev Tech Software Engineer- Pro Vis

20 years in the industry

Joined NVIDIA in March 2015.

Real time graphics makes me happy

I also like helicopters

Chris Hebert - Circa 1974

Agenda

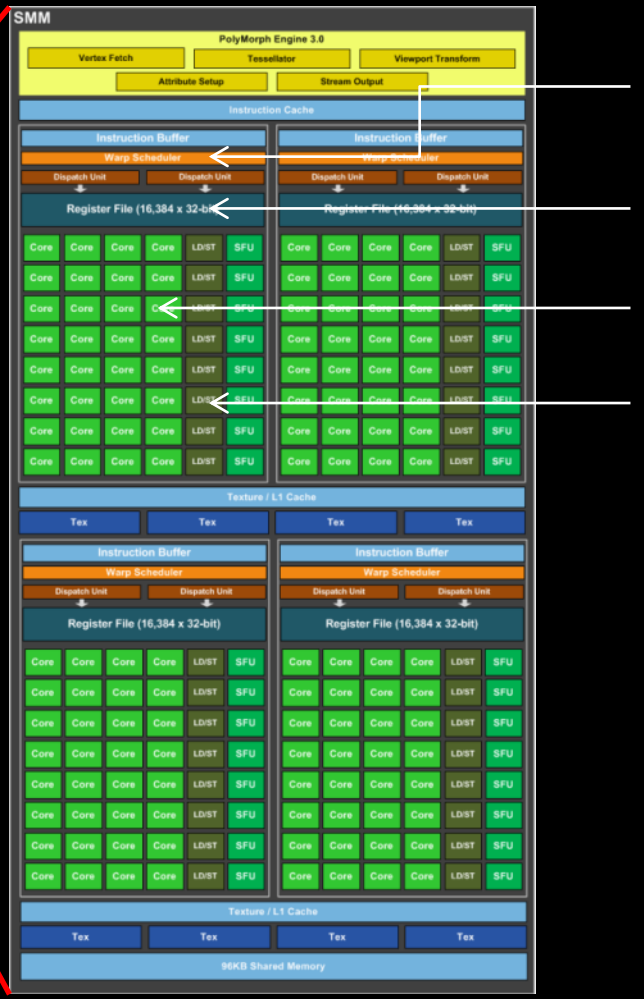
- Some Context
- Sharing The Load
- Pipeline Barriers

Some Context

GPU Architecture

In a nutshell

NVIDIA Maxwell 2



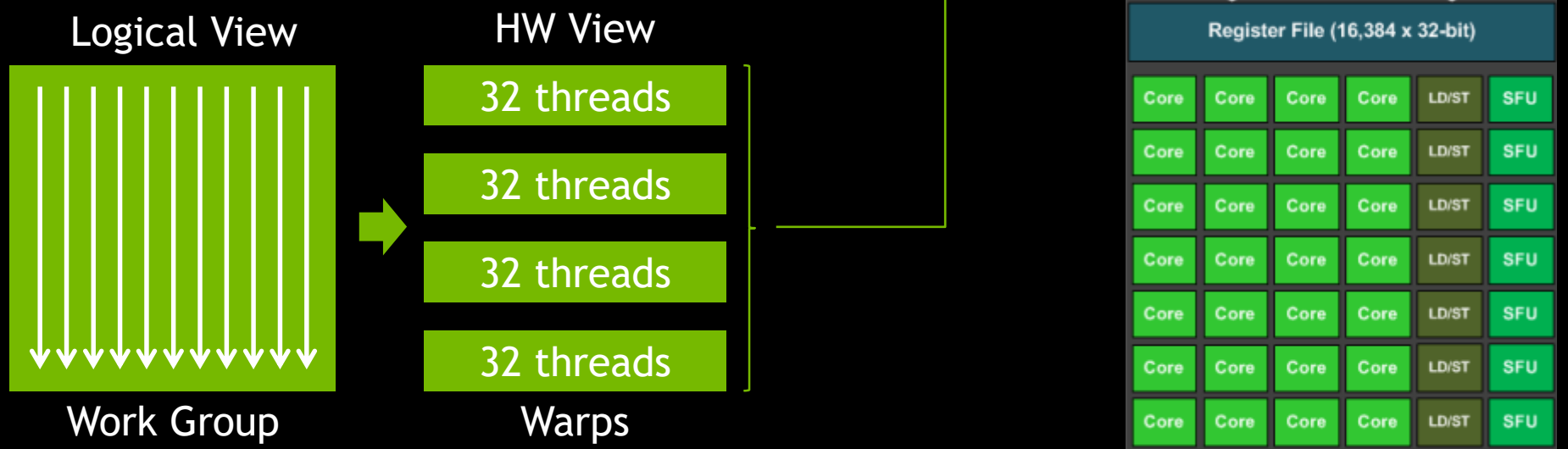
Register File

Core

Load Store Unit

Execution Model

Thread Hierarchies



Resource Partitioning

Resources Are Limited

Key resources impacting local execution:

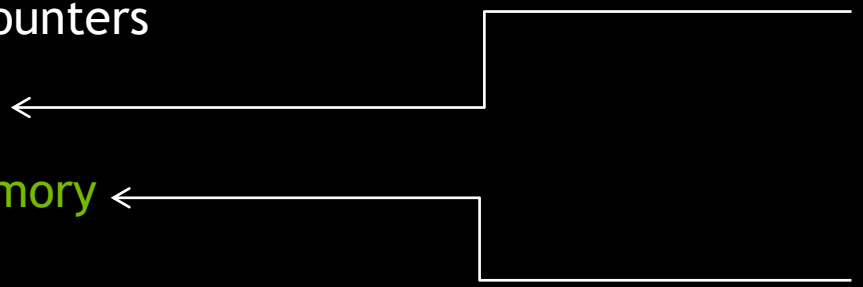
- Program Counters
- Registers
- Shared Memory

Resource Partitioning

Resources Are Limited

Key resources impacting local execution:

- Program Counters
- Registers
- Shared Memory



Partitioned amongst threads

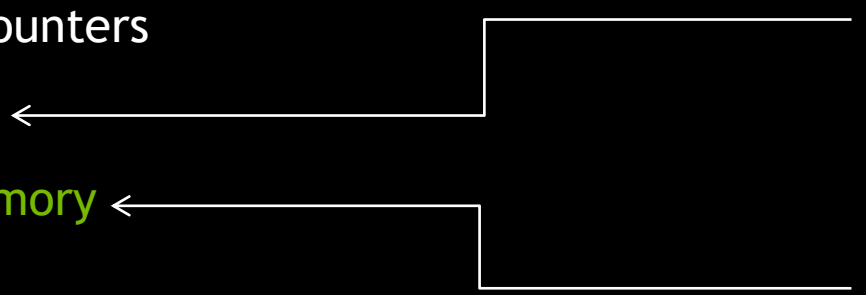
Partitioned amongst work groups

Resource Partitioning

Resources Are Limited

Key resources impacting local execution:

- Program Counters
- Registers
- Shared Memory



Partitioned amongst threads

Partitioned amongst work groups

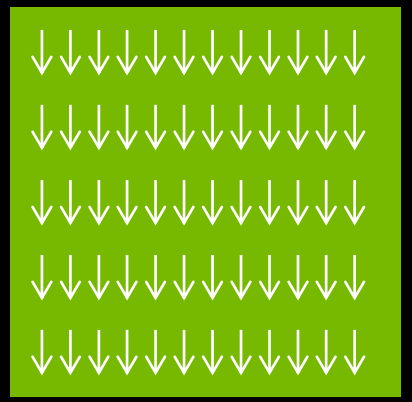
e.g. GTX 980 ti
64k 32bit registers per SM
96kb shared memory per SM

Resource Partitioning

Registers

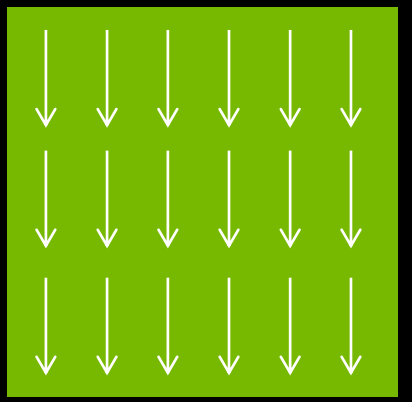
The more registers used by a kernel means few resident warps on the SM

Fewer Registers



More Threads

More Registers



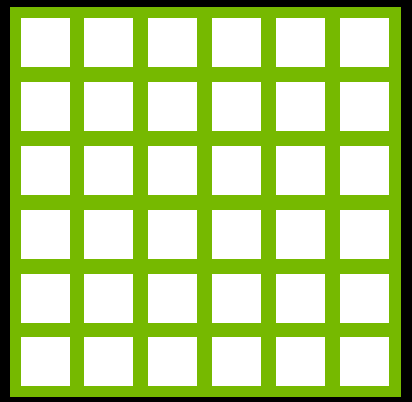
Fewer Threads

Resource Partitioning

Shared Memory

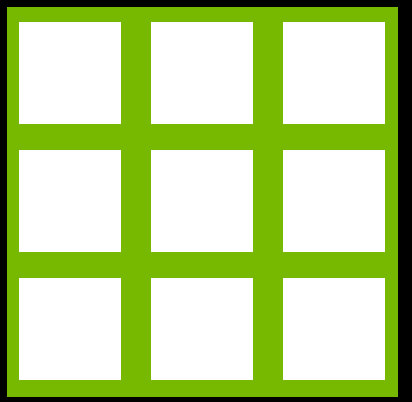
The more shared memory used by a work group means fewer work groups on the SM

Less SMEM



More Groups

More SMEM



Fewer Groups

Keeping It Moving
Occupancy

- Some small kernels may have low occupancy
  - Depending on the algorithm
- Compute resources are limited
  - Shared across threads or work groups on a per SM basis
- Warps stall when they have to wait for resources
- This latency can be hidden
  - If there are other warps ready to execute.

Keeping It Moving

Occupancy – Simple Theoretical Example

- Simple kernel that updates positions of 20480 particles
  - 1 FMAD - ~20 cycles (instruction latency)
  - 20480 particles = 640 warps
  - To hide this latency, according to Littles Law
    - Required Warps = Latency x Throughput
    - Throughput should be 32 threads * 16 sms = 512 to keep GPU busy
    - Required warps is 20*512 = 10240
    - ….oh….

Keeping It Moving

Occupancy – Simple Theoretical Example

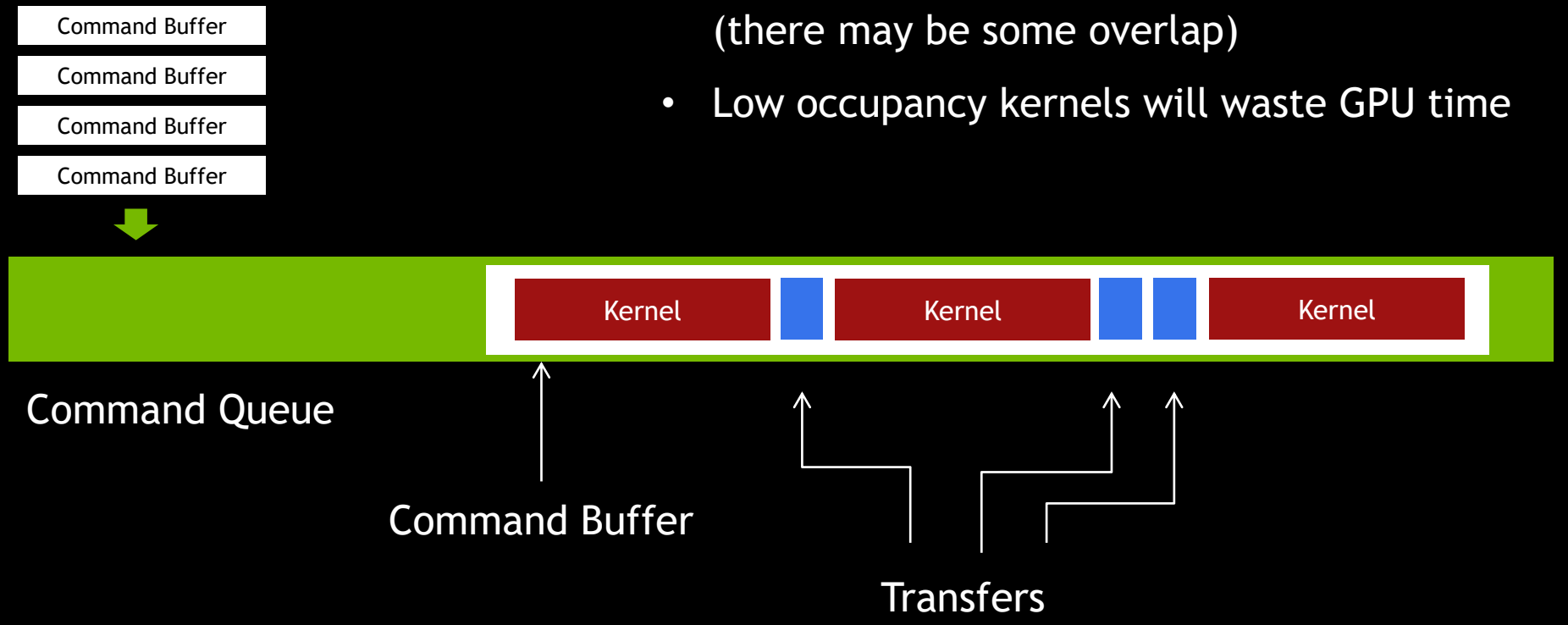
- Simple kernel that updates positions of 20480 particles
  - 1 FMAD - ~20 cycles (instruction latency)
  - 20480 particles = 640 warps
  - To hide this latency, according to Littles Law – But only on 1 SM..
    - Required Warps = Latency x Throughput
    - Throughput should be 32 threads * 1 sm = 32 to keep GPU busy
    - Required warps is 20*32 = 640
    - And we theoretically have 15 SMs to use for other stuff.

Queuing It Up

Working with 1 Queue

- Scheduler will distribute work across all SMs
- kernels execute in sequence

  (there may be some overlap)
- Low occupancy kernels will waste GPU time



Sharing The Load

Queuing It Up

Working with N Queues

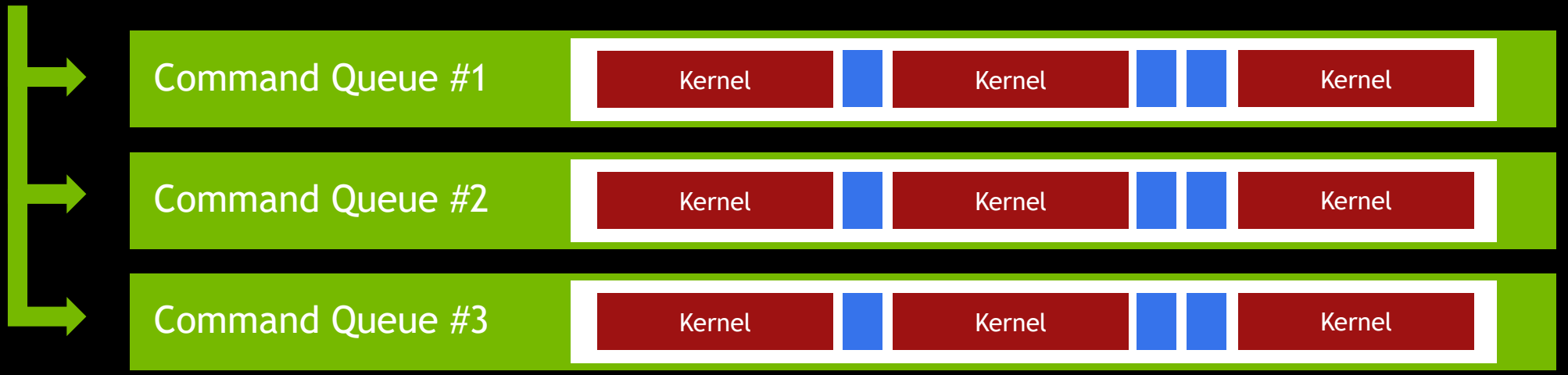
Command Buffer

Command Buffer

Command Buffer

Command Buffer

- NVIDIA hardware gives you 16 all powerful queues
- 1 Queue family that supports all operations
- 16 queues available for use



Queuing It Up

Working with N Queues

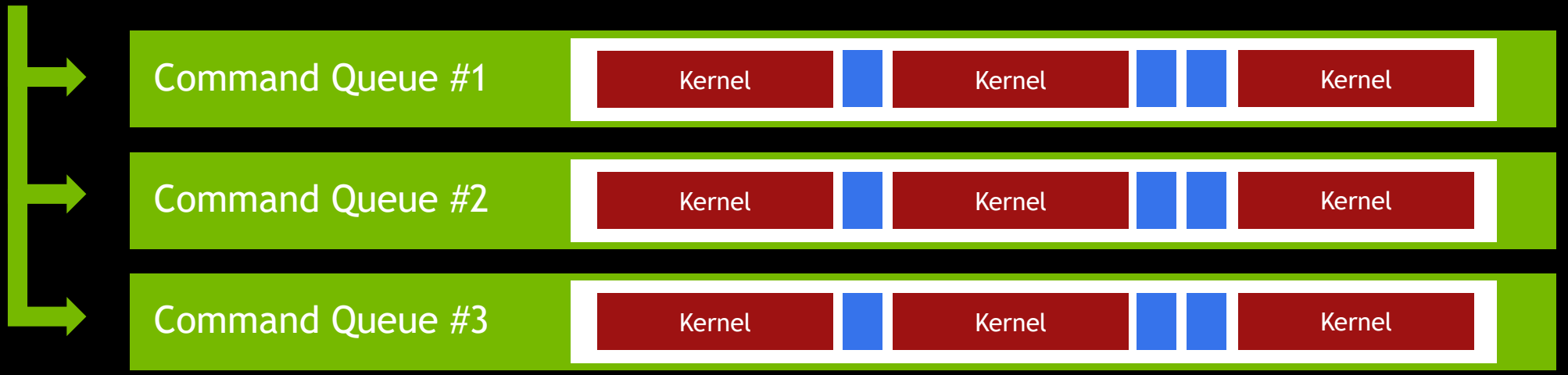
Command Buffer

Command Buffer

Command Buffer

Command Buffer

- Application decides which queues for which kernels
- Load balance for best performance
- Profile (Nsight) to gain insights



Queuing It Up

Compute and Graphics In Harmony

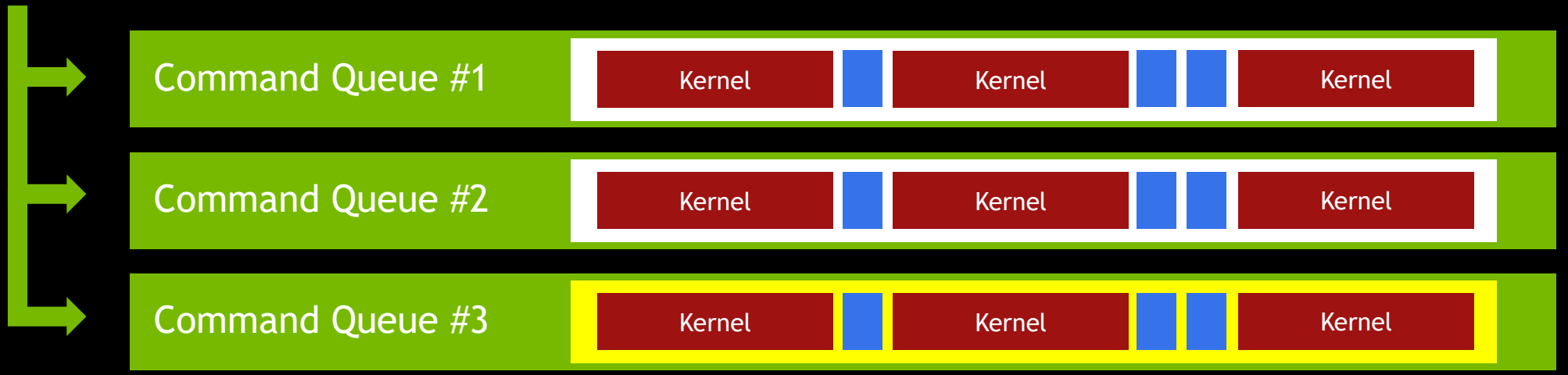
- Some hardware can even run compute and graphics work concurrently
- Needs fast context switching and at high granularity (not just at draw commands)
- Simple Graphics work tends to have high occupancy
- Complex graphics work can reduce occupancy
- Profile for performance insights

Queuing It Up

Compute and Graphics In Harmony

Compute Cmd Buffer
Compute Cmd Buffer
Graphics Cmd Buffer
Compute Cmd Buffer

- Profile to understand occupancy of both graphics and compute workloads
- Queues can support both compute and graphics



An Example

Compute and Graphics In Harmony

Click here to view this video

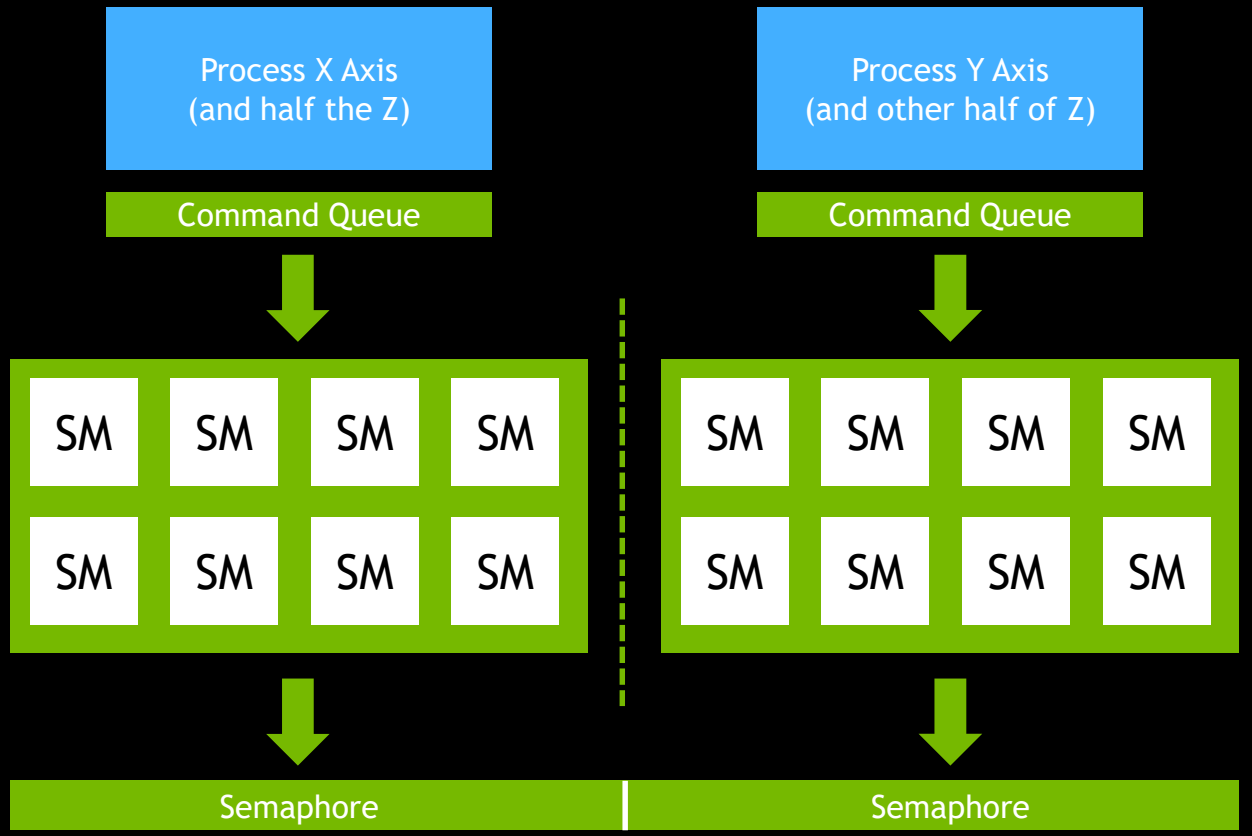
Free Surface Navier Stokes Solver

- 11 Compute Kernels
- 4 Shaders

- The output of each kernel is the input to the next
- Some kernels have very low occupancy
- Still opportunities for concurrency with compute

An Example

Many discretized operations are separable



Examples

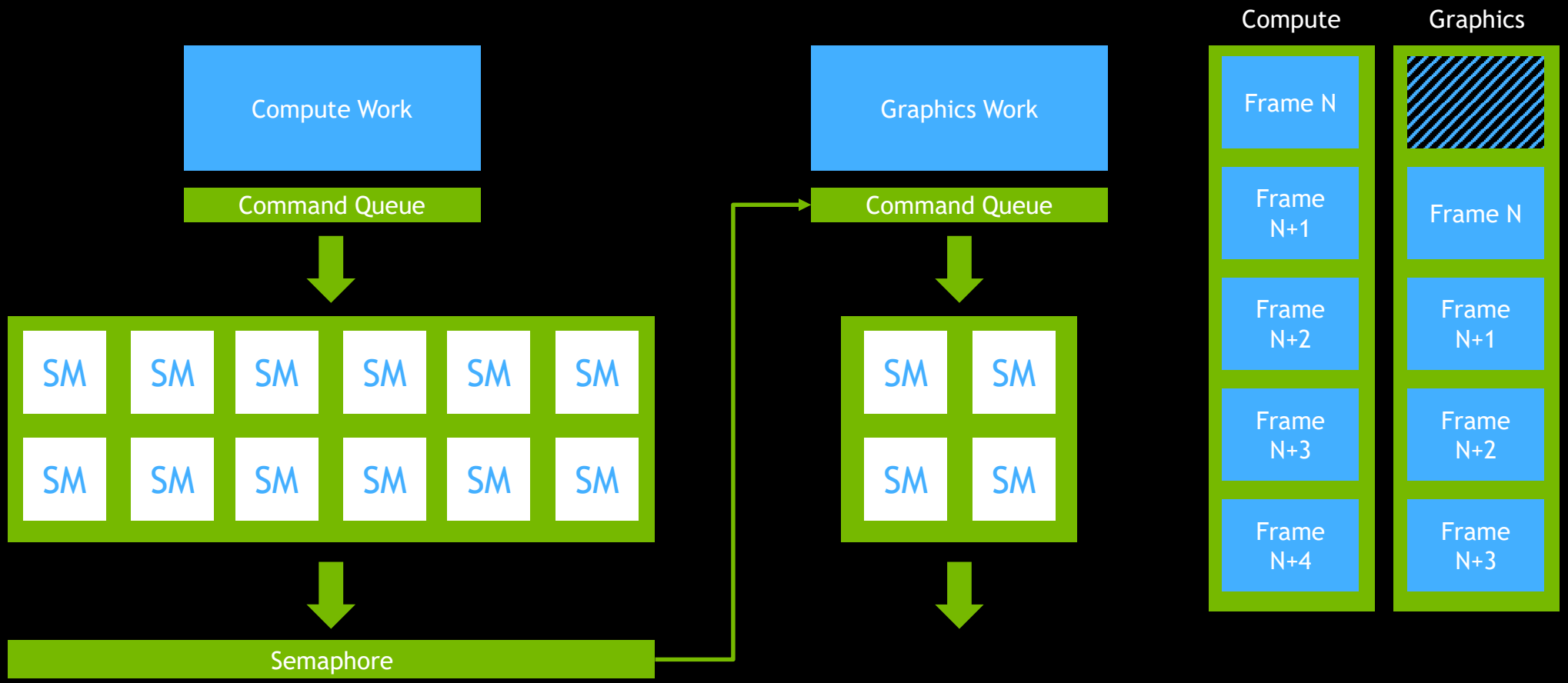
- Fluid Sims
- Gaussian Blurs
- Convolution Kernels

Driver handles dispatching groups

Use semaphores to synchronize

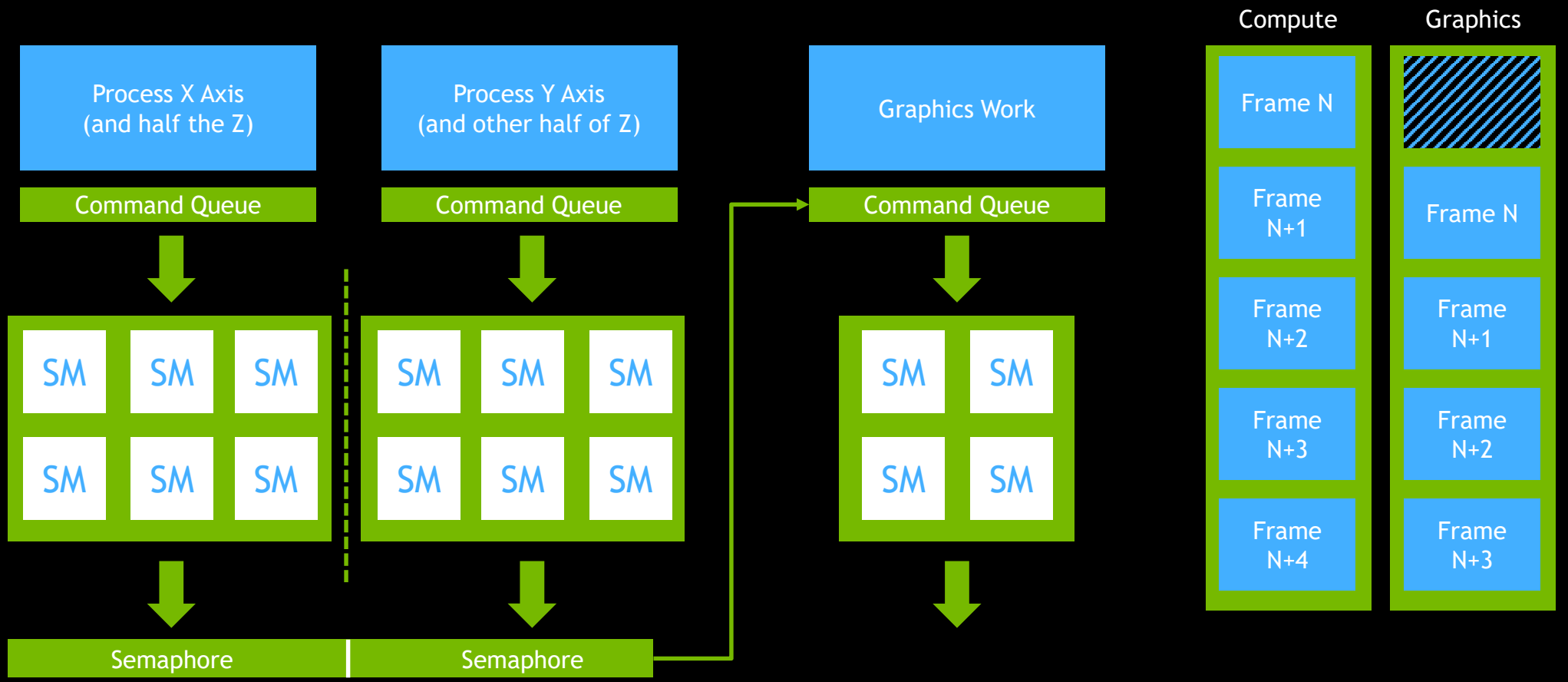
An Example

Compute and graphics run concurrently



An Example

Putting it all together

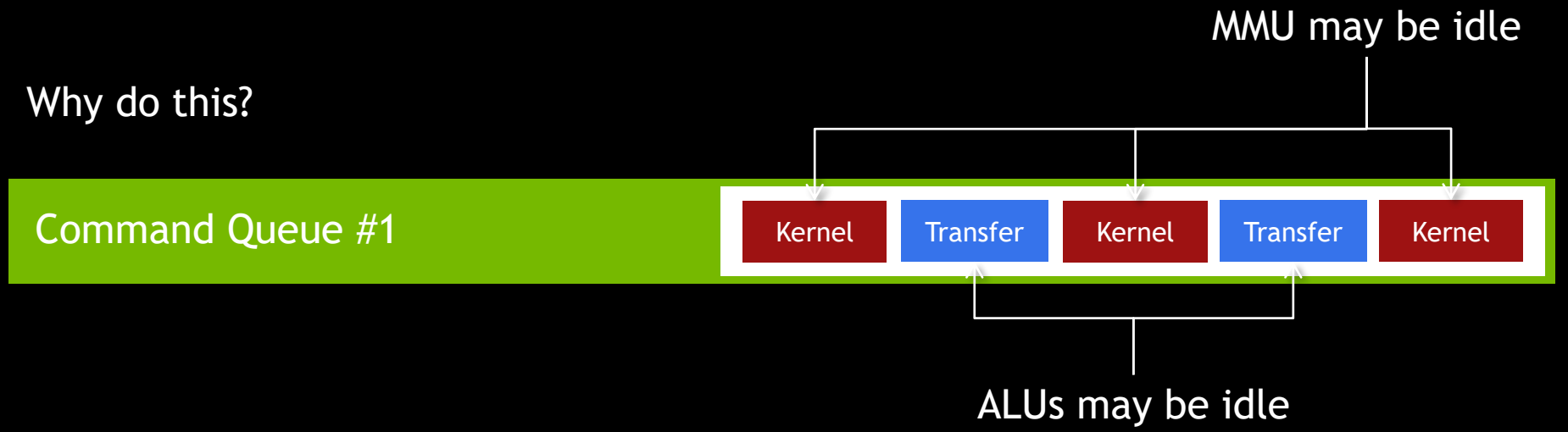


Memory Transfers

More opportunity for concurrency

- Memory transfers are handle by MMU
- Can run concurrently with Kernels
  - As long as the current kernel isnt using the memory

Why do this?

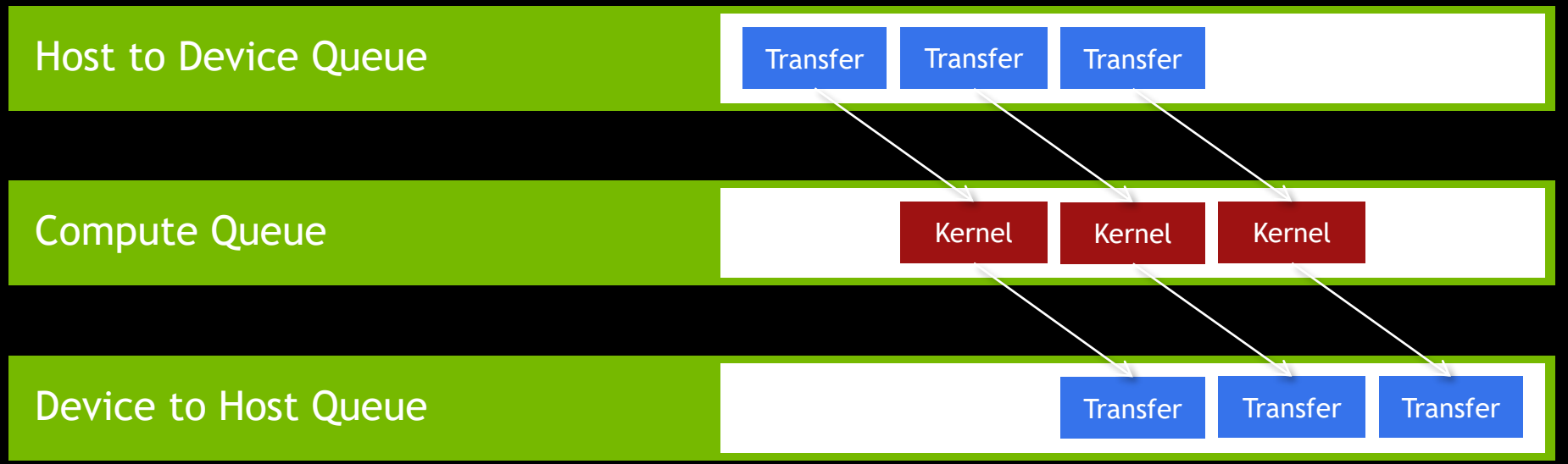


Memory Transfers

More opportunity for concurrency

When you can do this
- DtoH and HtoD transfers can run concurrently

Examples
- Large image processing
- Video processing



Conclusion

Takeaways

There is more than 1 queue available

Keep registers and shared memory to a minimum

Low occupancy leads to an under utilized GPU

Maximize GPU utilization by running kernels concurrently

Profile to understand the occupancy profiles of kernels and shaders

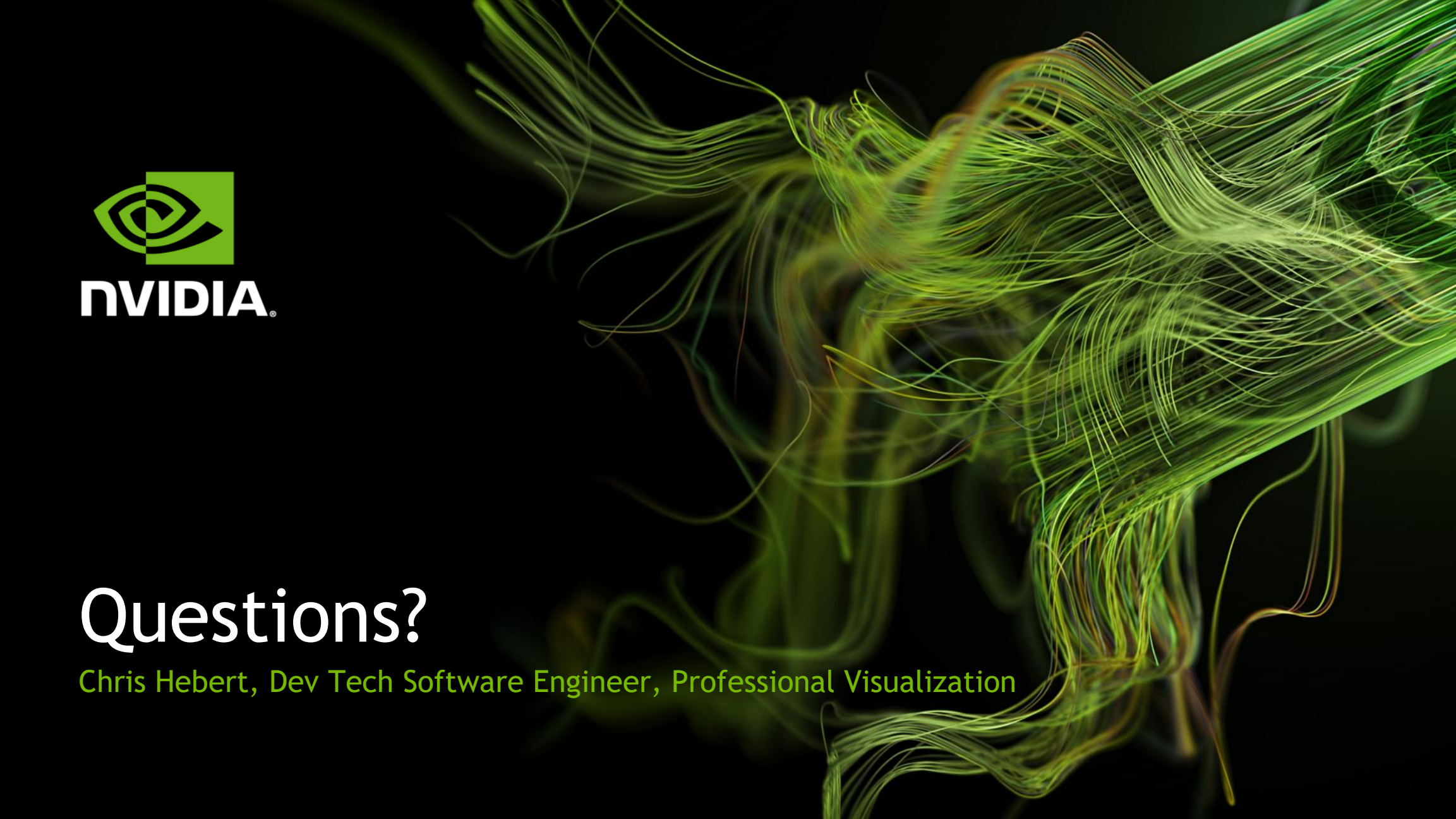
Some hardware can run kernels AND shaders concurrently

Use Semaphores to synchronize between queues

Be sensible at the beer festival

Thank You

Enjoy Vulkan!!



Questions?

Chris Hebert, Dev Tech Software Engineer, Professional Visualization