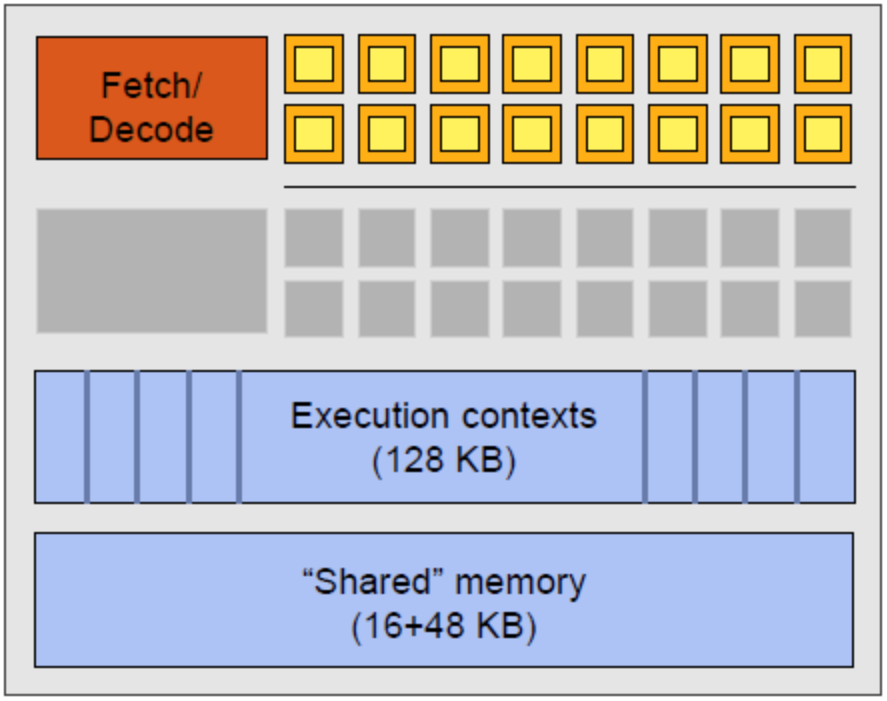
Topic for today:

Graphical Processing Units (GPUs)

Graphical Processing Unit

A GPU is a heterogeneous chip multi-processor (highly tuned for graphics)



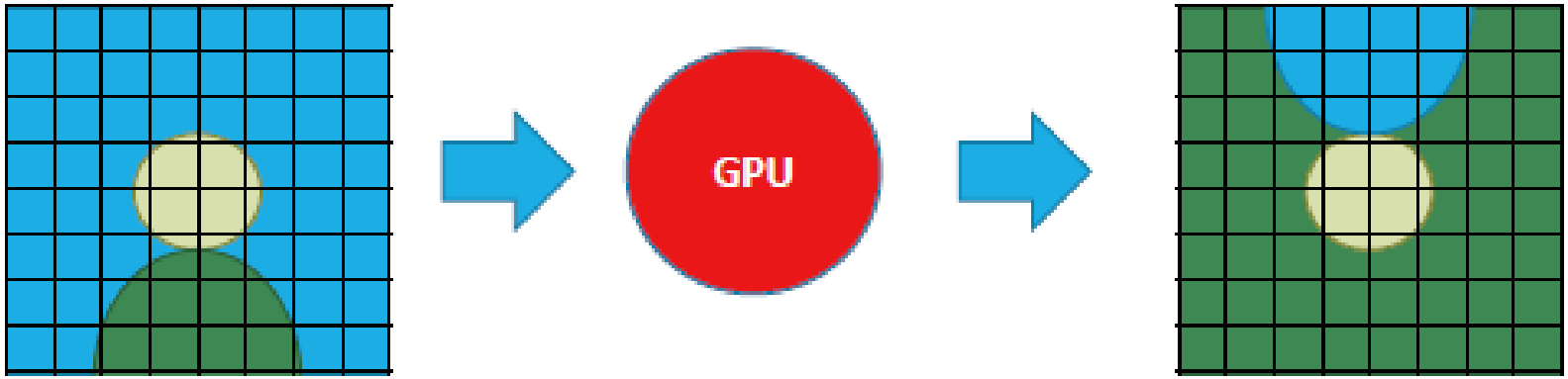
Source: Fermi Compute Architecture Whitepaper
CUDA Programming Guide 3.1, Appendix G

GPU-accelerated computing

Use of a graphics processing unit (GPU) together with a CPU to accelerate scientific, analytics, engineering, consumer, and enterprise applications.

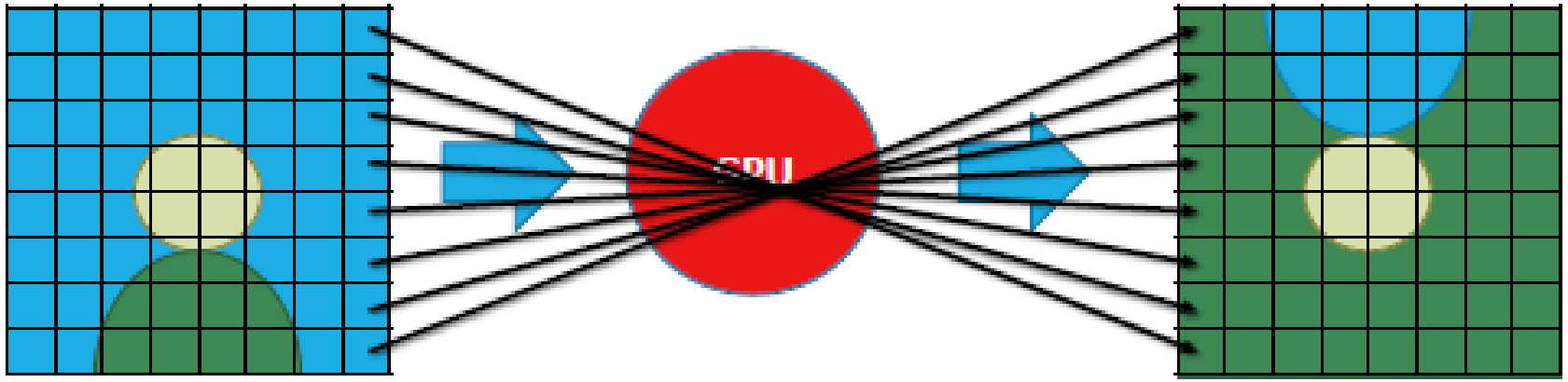
Graphics Workload

Streaming computation *on pixels*



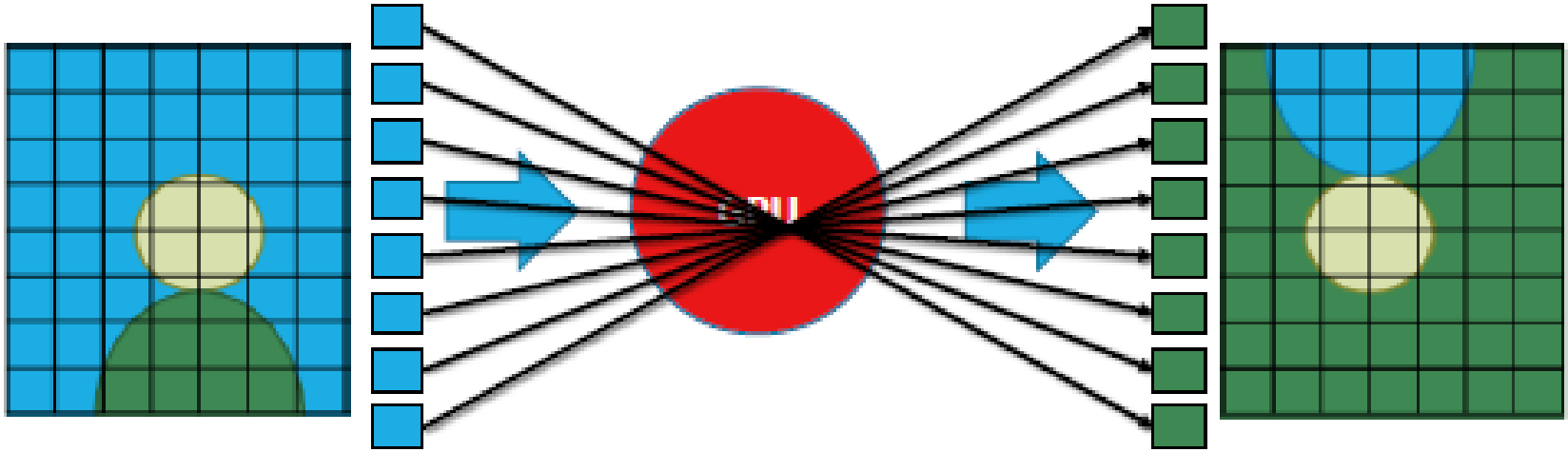
Graphics Workload

Identical, Streaming computation *on pixels*



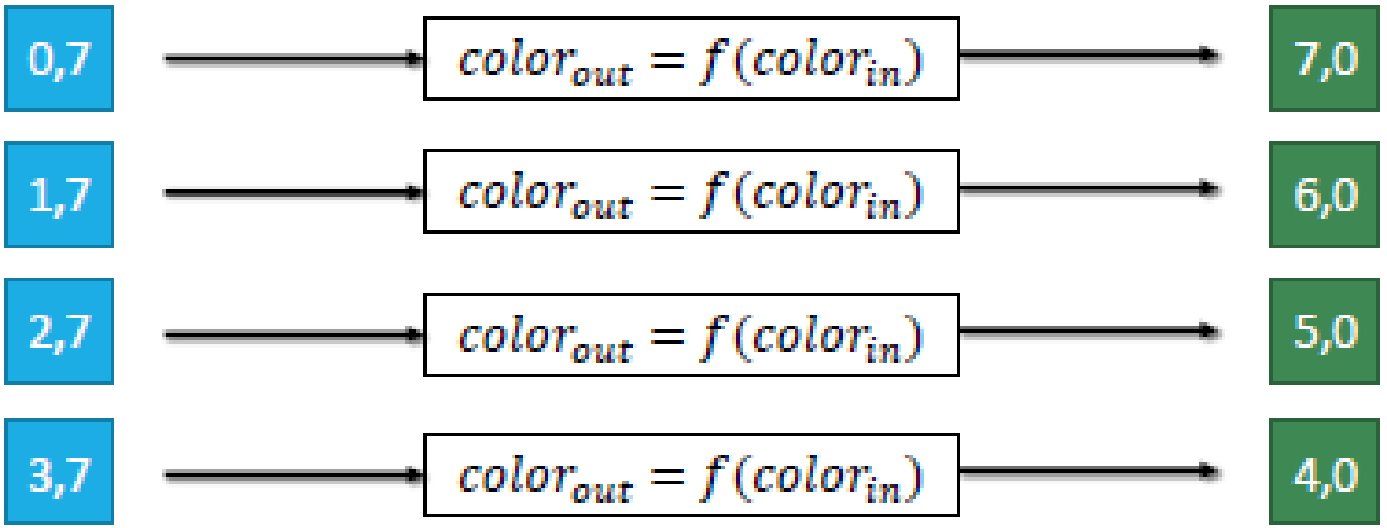
Graphics Workload

Identical, Independent, Streaming computation *on pixels*



Generalize: Data Parallel Workloads

Identical, Independent computation *on multiple data inputs*



Programming the GPU (NVIDIA)

- Heterogeneous execution model
  - CPU is the *host*, GPU is the *device*
- Develop a C-like programming language for GPU
- Unify all forms of GPU parallelism as *CUDA thread*
- Programming model is “Single Instruction Multiple Thread”

Threads and Blocks

- A thread is associated with each data element
- Threads are organized into blocks
- Blocks are organized into a grid

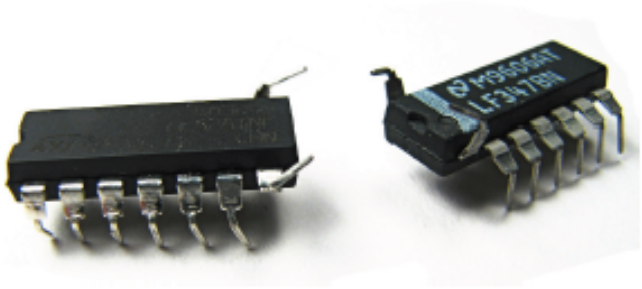
- GPU hardware handles thread management, not applications or OS

NVIDIA GPU Architecture

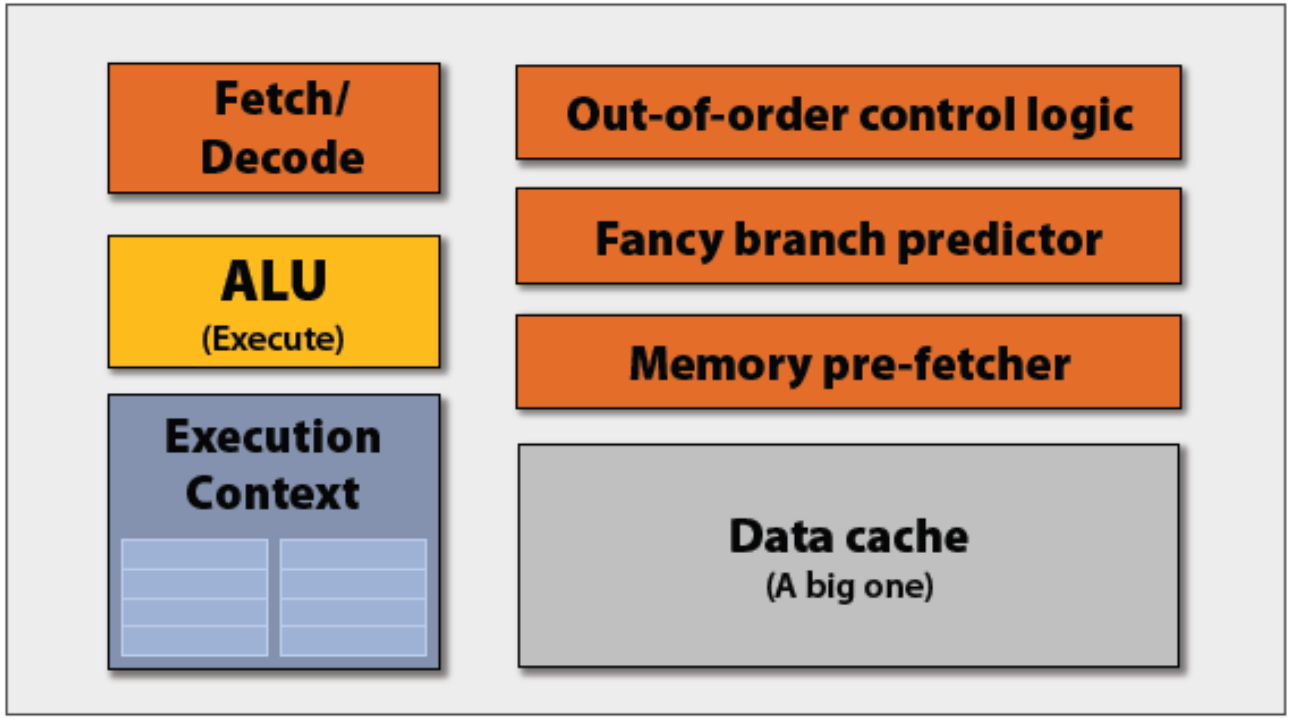
- Similarities to vector machines:
  - Works well with data-level parallel problems
  - Scatter-gather transfers
  - Mask registers
  - Large register files
- Differences:
  - No scalar processor
  - Uses multithreading to hide memory latency
  - Has many functional units, as opposed to a few deeply pipelined units like a vector processor

Why GPU?

- Design target for CPUs:
  - Make a single thread very fast
  - Take control away from programmer
- GPU Computing takes a different approach:
  - Throughput matters—single threads do not
  - Give explicit control to programmer

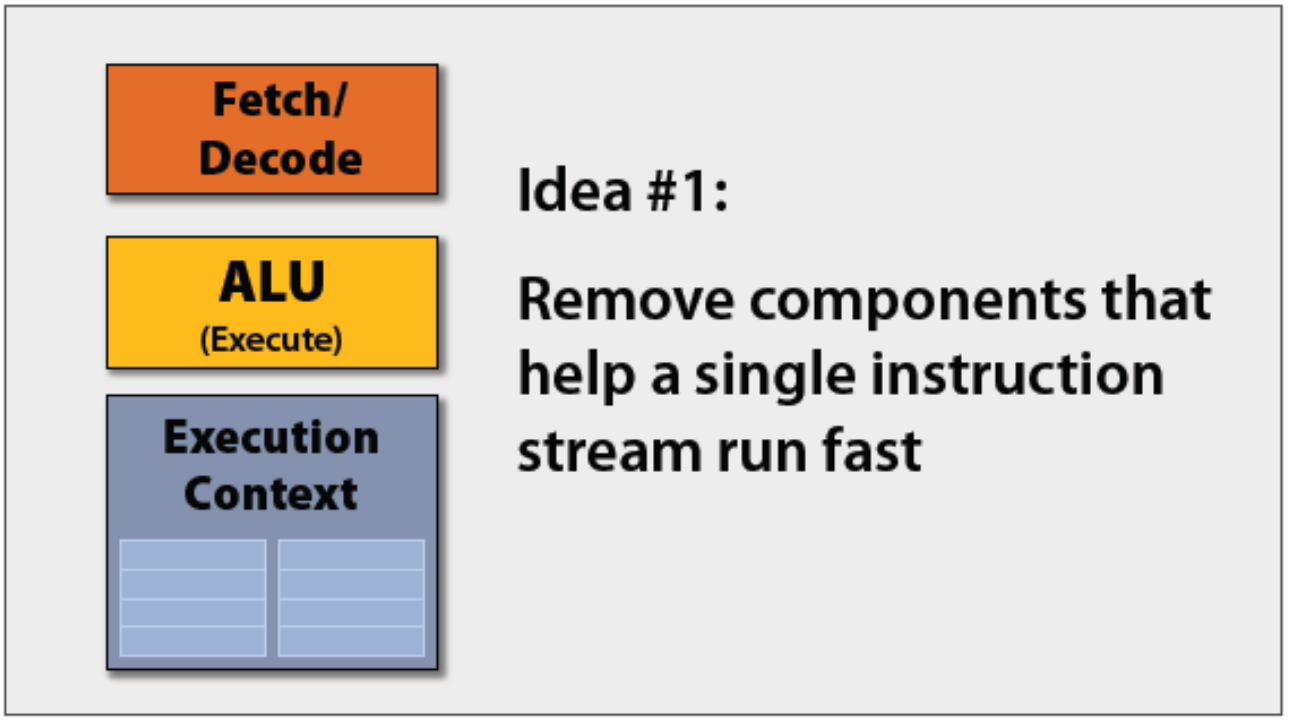


"CPU-style" Cores



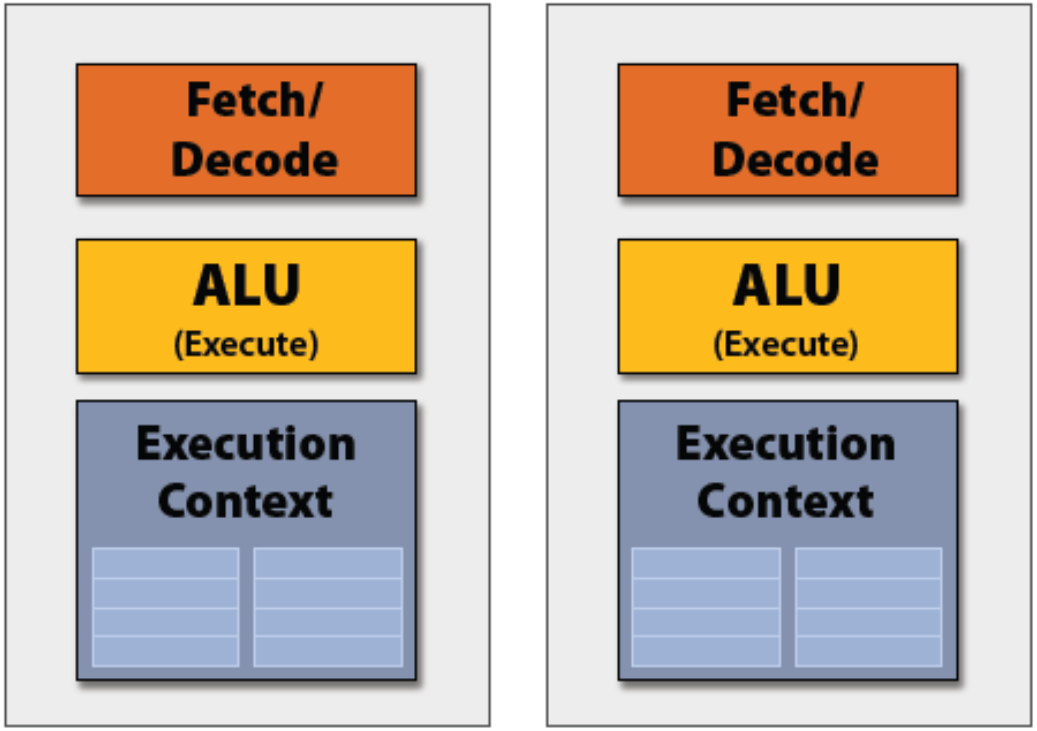
Credit: Kayvon Fatahalian (Stanford)

Slimming down



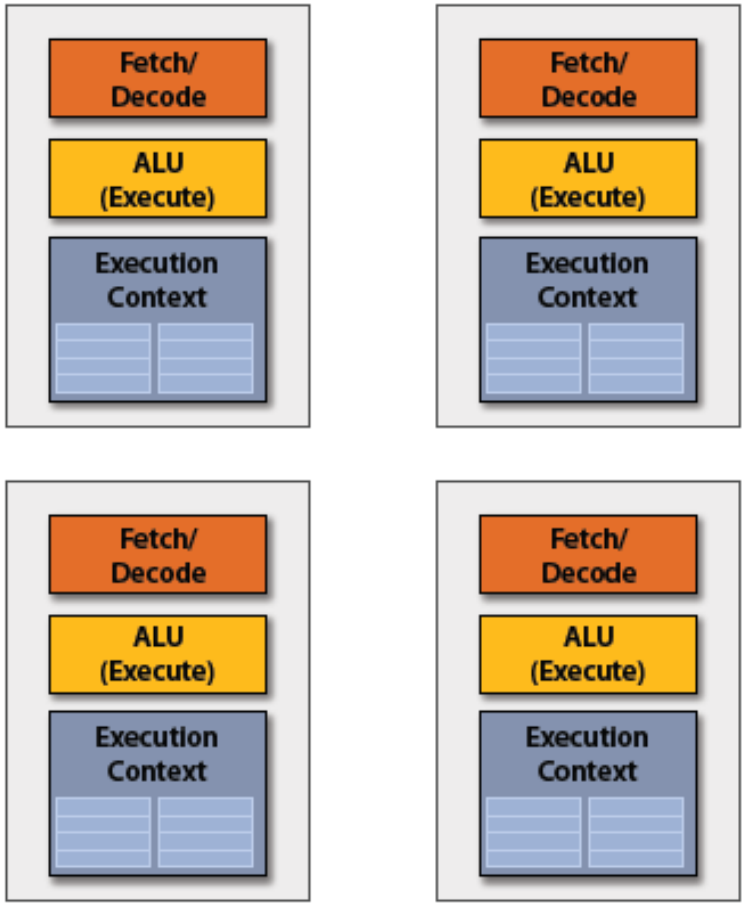
Credit: Kayvon Fatahalian (Stanford)

More Space: Double the Number of Cores



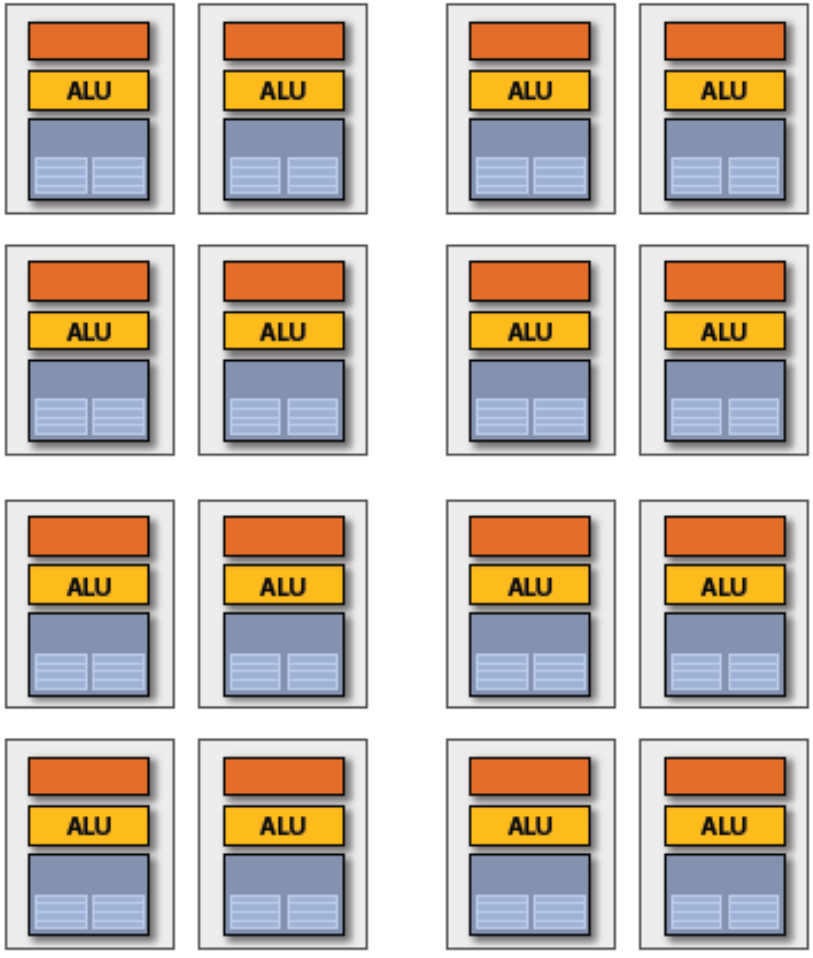
Credit: Kayvon Fatahalian (Stanford)

. . . again



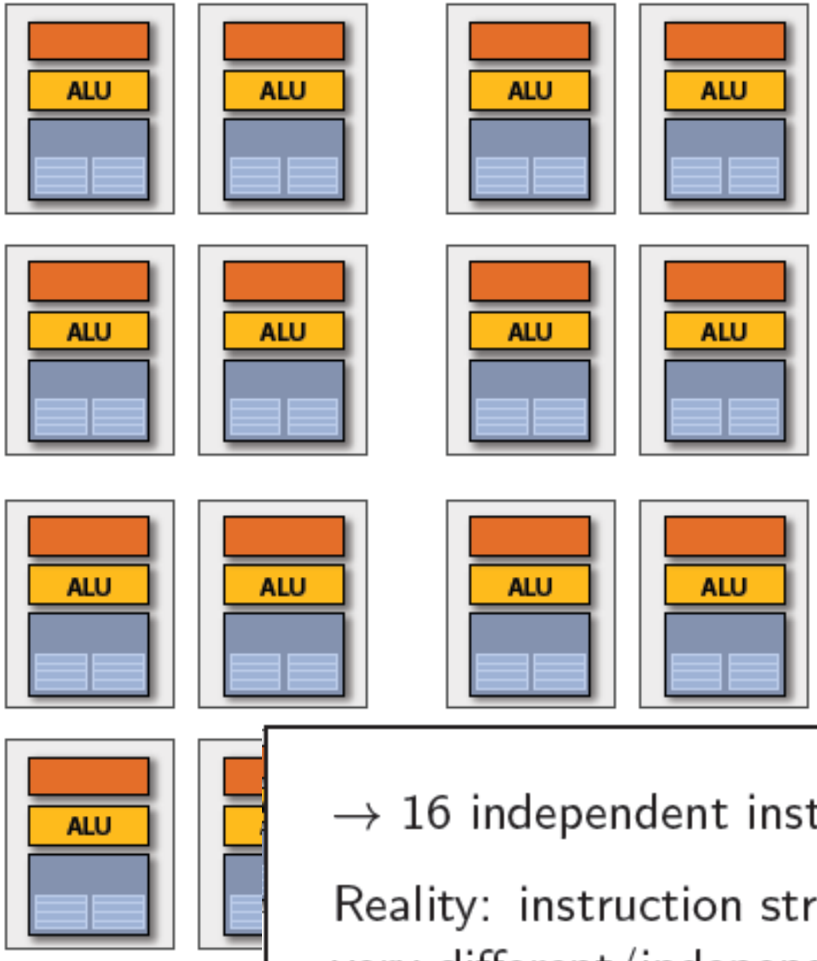
Credit: Kayvon Fatahalian (Stanford)

. . . and again



Credit: Kayvon Fatahalian (Stanford)

. . . and again

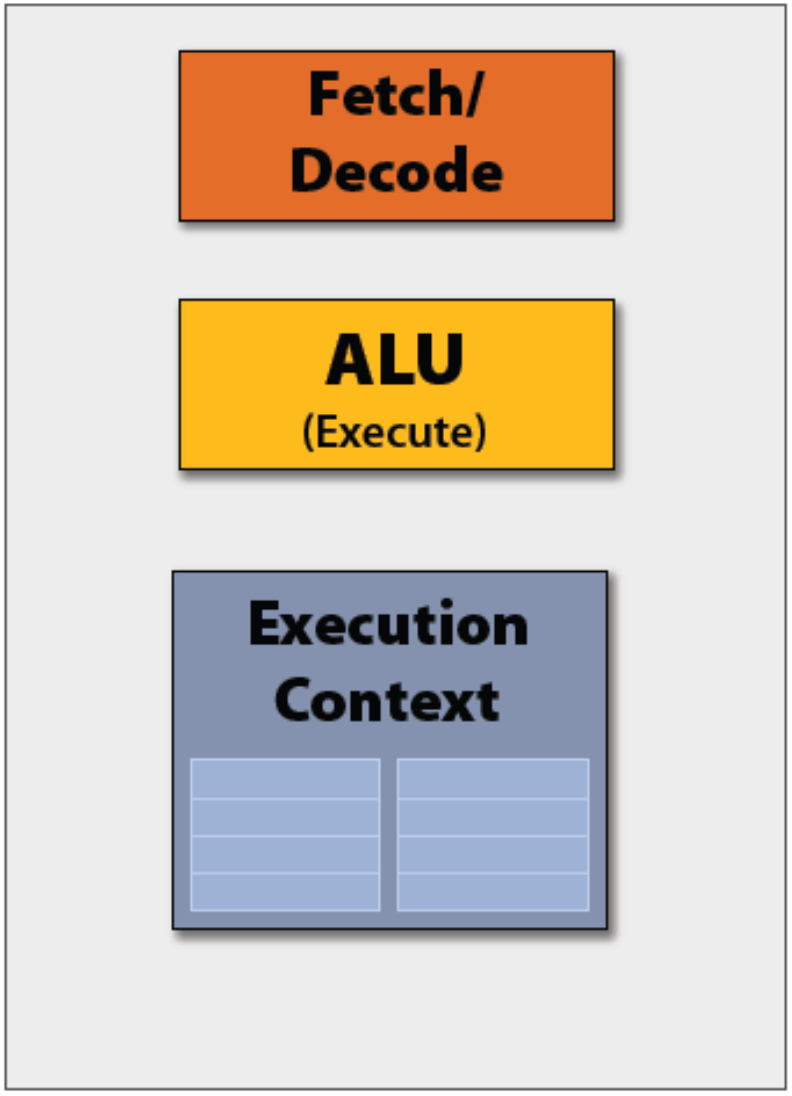


→ 16 independent instruction streams

Reality: instruction streams not actually very different/independent

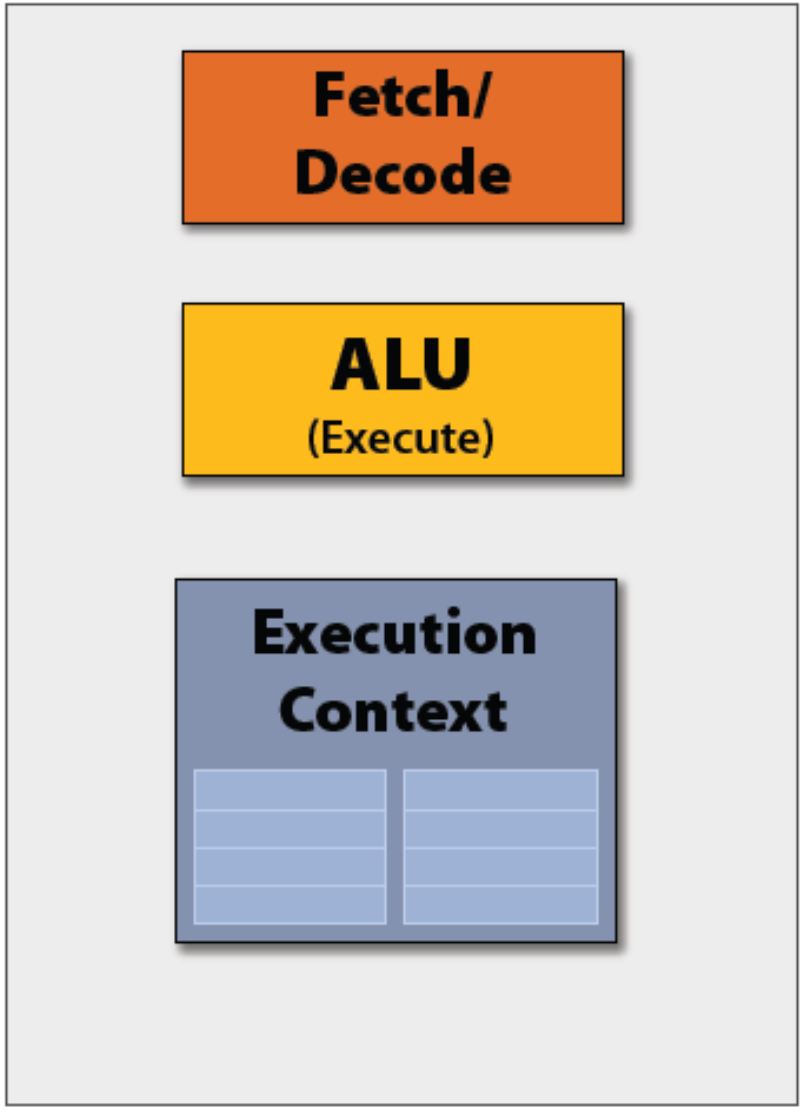
Credit: Kayvon Fatahalian (Stanford)

Saving Yet More Space



Credit: Kayvon Fatahalian (Stanford)

Saving Yet More Space



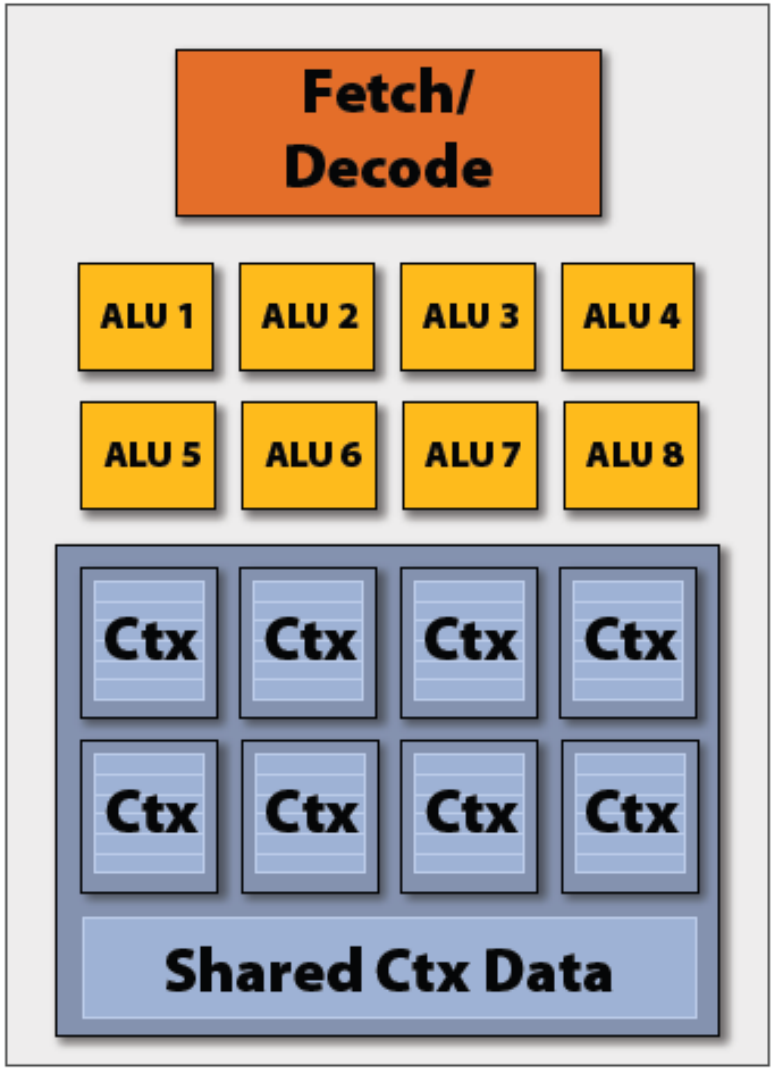
Idea #2

Amortize cost/complexity of managing an instruction stream across many ALUs

→ **SIMD**

Credit: Kayvon Fatahalian (Stanford)

Saving Yet More Space



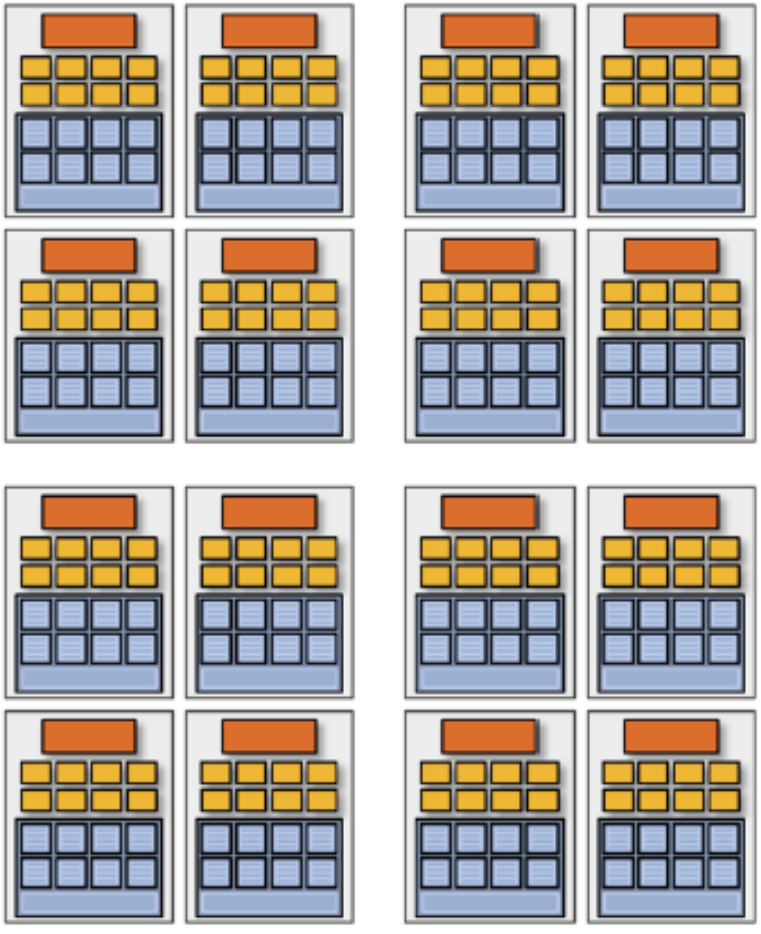
Idea #2

Amortize cost/complexity of managing an instruction stream across many ALUs

→ **SIMD**

Credit: Kayvon Fatahalian (Stanford)

Gratuitous Amounts of Parallelism!

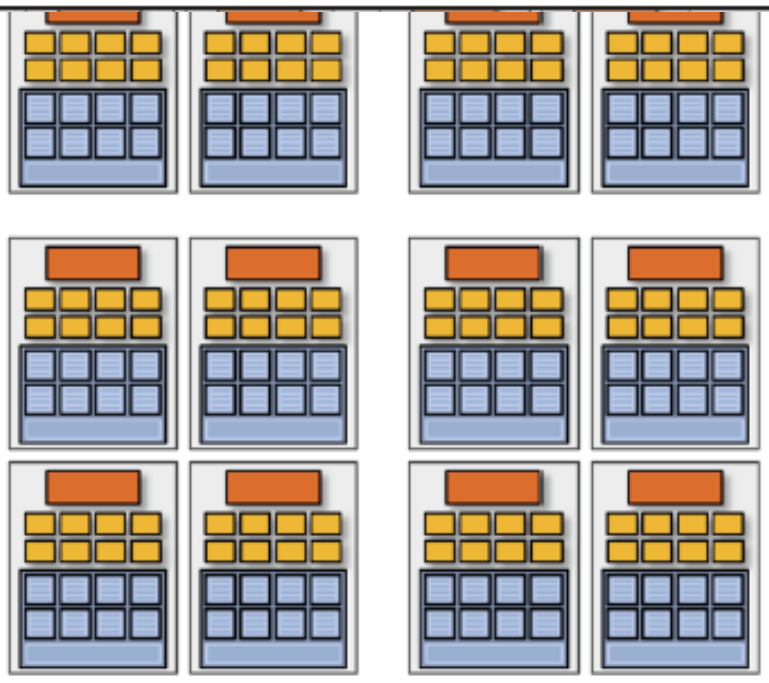


Credit: Kayvon Fatahalian (Stanford)

Gratuitous Amounts of Parallelism!

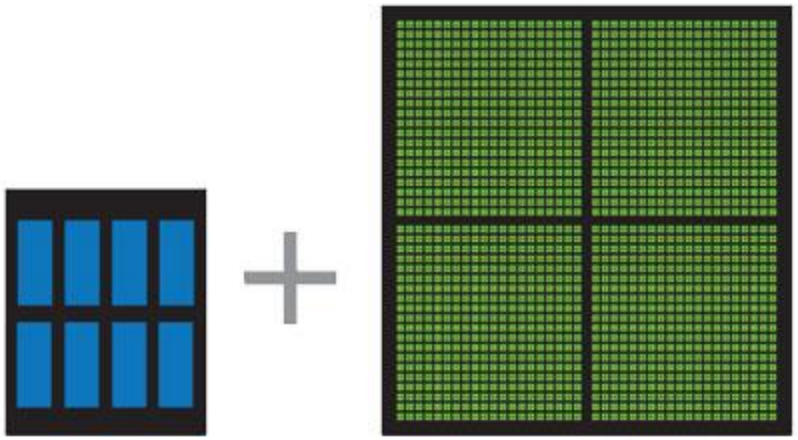
Example:

128 instruction streams in parallel
16 independent groups of 8 synchronized streams



Credit: Kayvon Fatahalian (Stanford)

CPU vs. GPU



https://www.youtube.com/watch?v=-P28LKWTzrI

What is CUDA?

- Scalable parallel programming model and a software environment for parallel computing
  - Minimal extensions to C/C++ environment
  - Heterogeneous serial-parallel programming model

What is OpenCL?

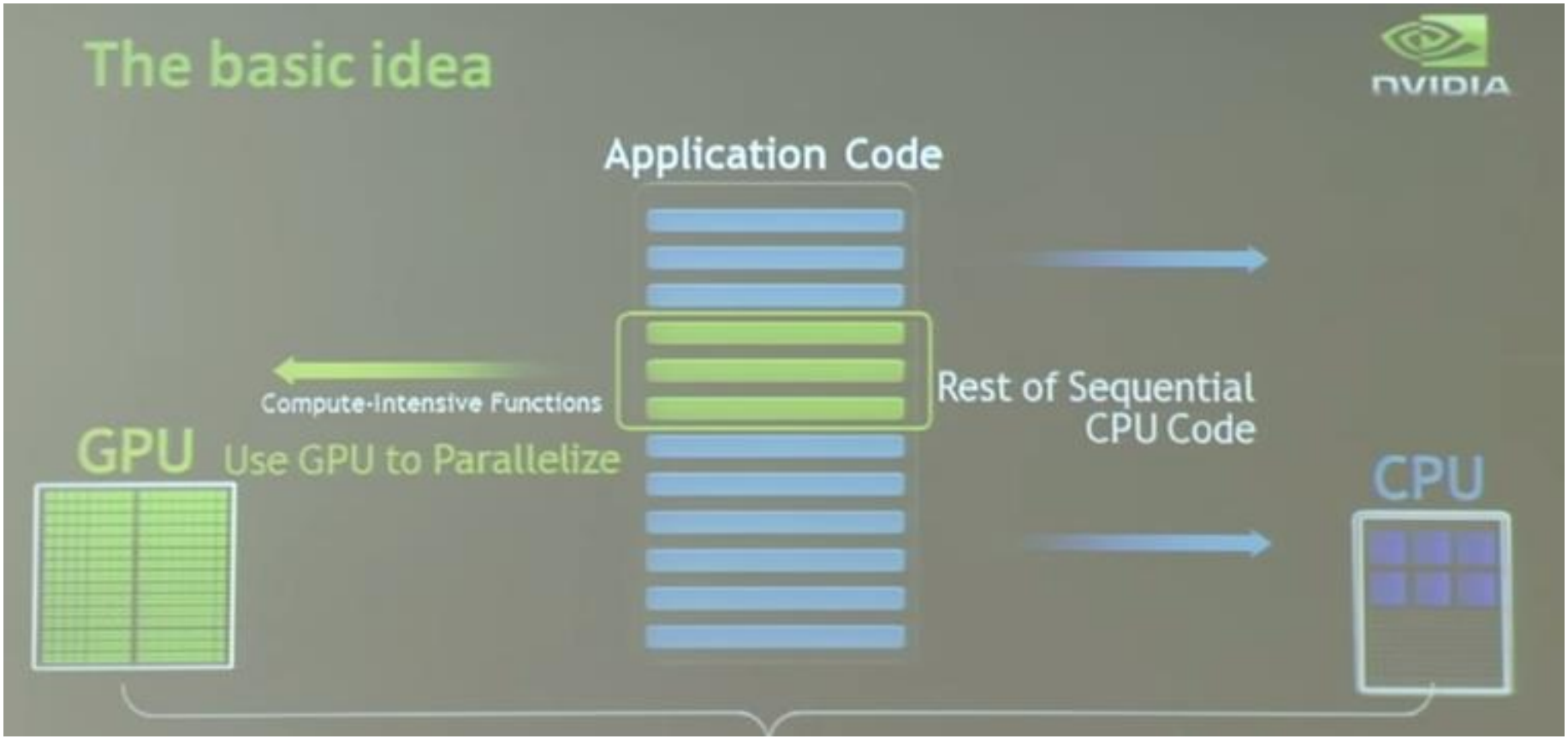
- OpenCL (Open Computing Language) is an open, royalty-free standard for general purpose parallel programming across CPUs, GPUs, and other processors.
  - Device neutral
  - Vendor neutral

GPGPU

GPU Computing: an emerging field seeking to harness GPUs for general-purpose computation.

Motivation: Flexible and Precise

- Modern GPUs are deeply programmable
  - Programmable pixel, vector, video engines
  - Solidifying high-level language support
- Modern GPUs support high precision
  - 32 bit floating point throughout the pipeline
  - High enough for many (not all) applications
  - Newest GPUs have 64 bit support


The basic idea
NVIDIA
Application Code
Compute-Intensive Functions
Rest of Sequential
CPU Code
GPU
Use GPU to Parallelize
CPU

Reducing Radiation from CT Scans

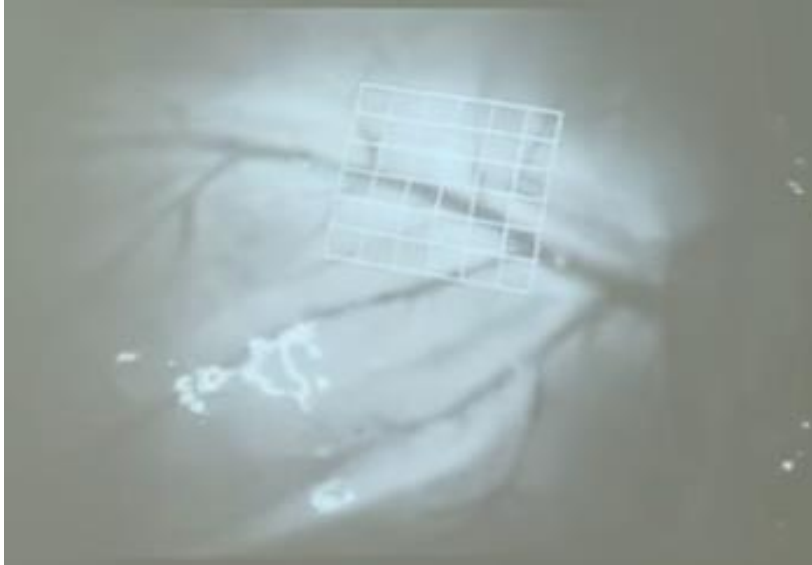
Est. 28,000 people/year get cancer from CT scans

UCSD: advanced CT reconstruction reduces radiation by **35-70x**

CPUs: 2 hours (unusable)

CUDA: 2 minutes (clinically practical)

Operating on a Beating Heart



Only 2% of surgeons will operate on a beating heart

Patient stands to lose 1 point of IQ every10 min with heart stopped

GPU enables real-time motion compensation to virtually stop beating heart for surgeons