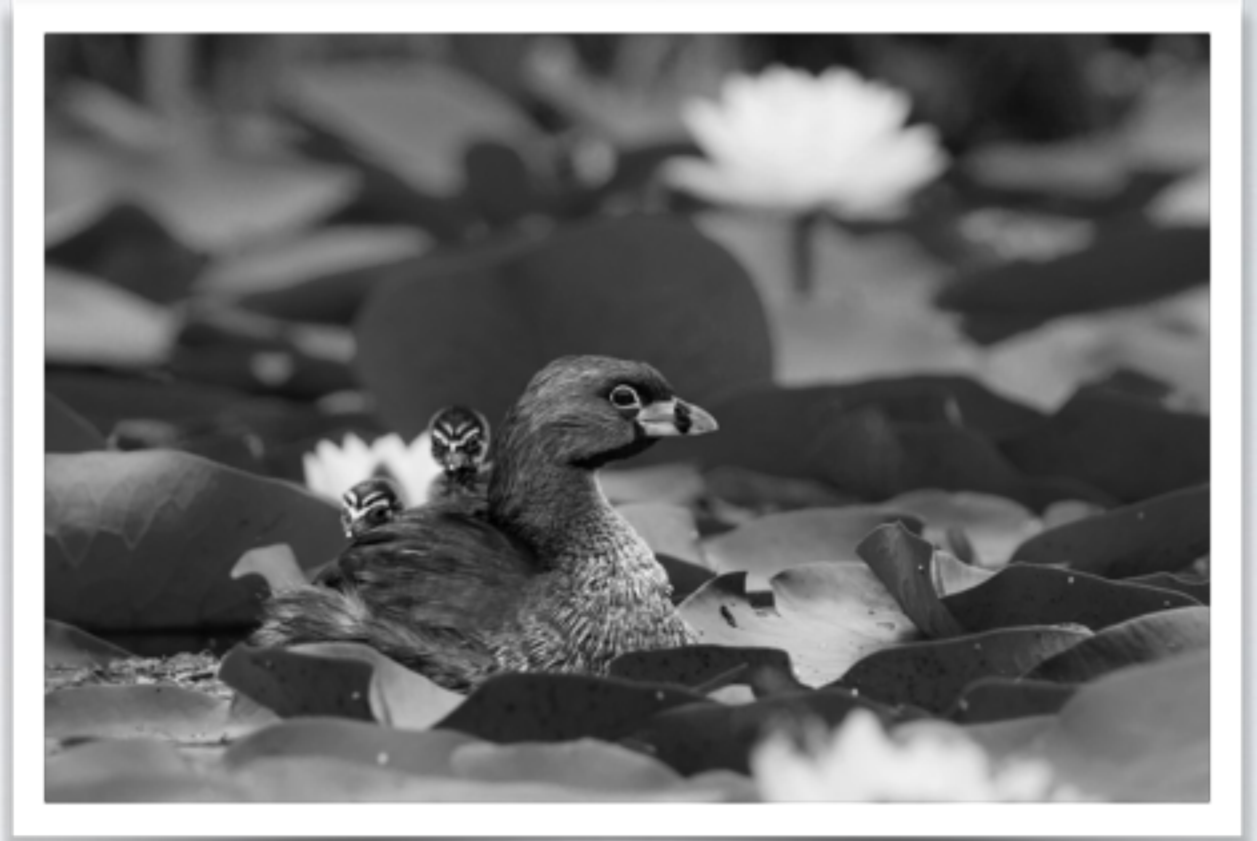
HOW TO CRUNCH NUMBERS ON GPU

for fun and profit (with CUDA, C and R)

"It all started with video games"

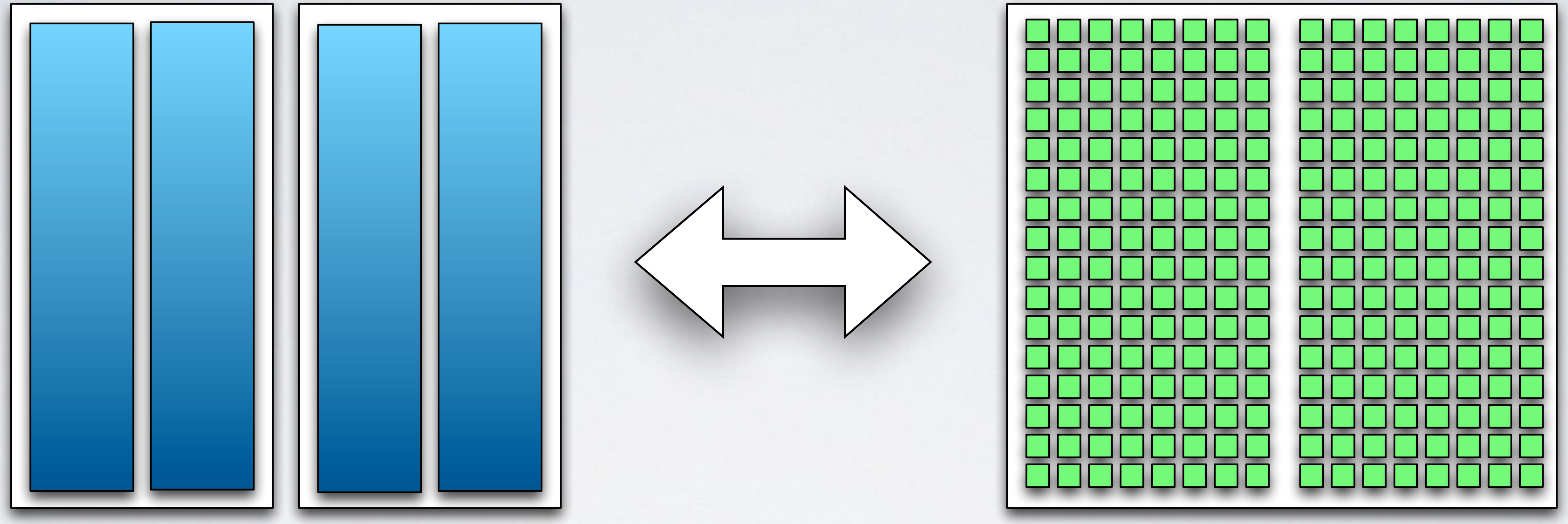
2013 Video Game Market: $93 Billion
2013 Business Analytics Market: $14 Billion

SHADERS



```
float4 Saturation : register(c0);
sampler2D implicitInputSampler : register(S0);

float4 main(float2 uv : TEXCOORD) : COLOR
{
  float3 LuminanceWeights = float3(0.299,0.587,0.114);
  float4 srcPixel = tex2D(implicitInputSampler, uv);
  float4 luminance = dot(srcPixel,LuminanceWeights);
  float4 dstPixel = lerp(luminance,srcPixel,Saturation);
  dstPixel.a = srcPixel.a;
  return dstPixel;
}
```

CPU vs GPU

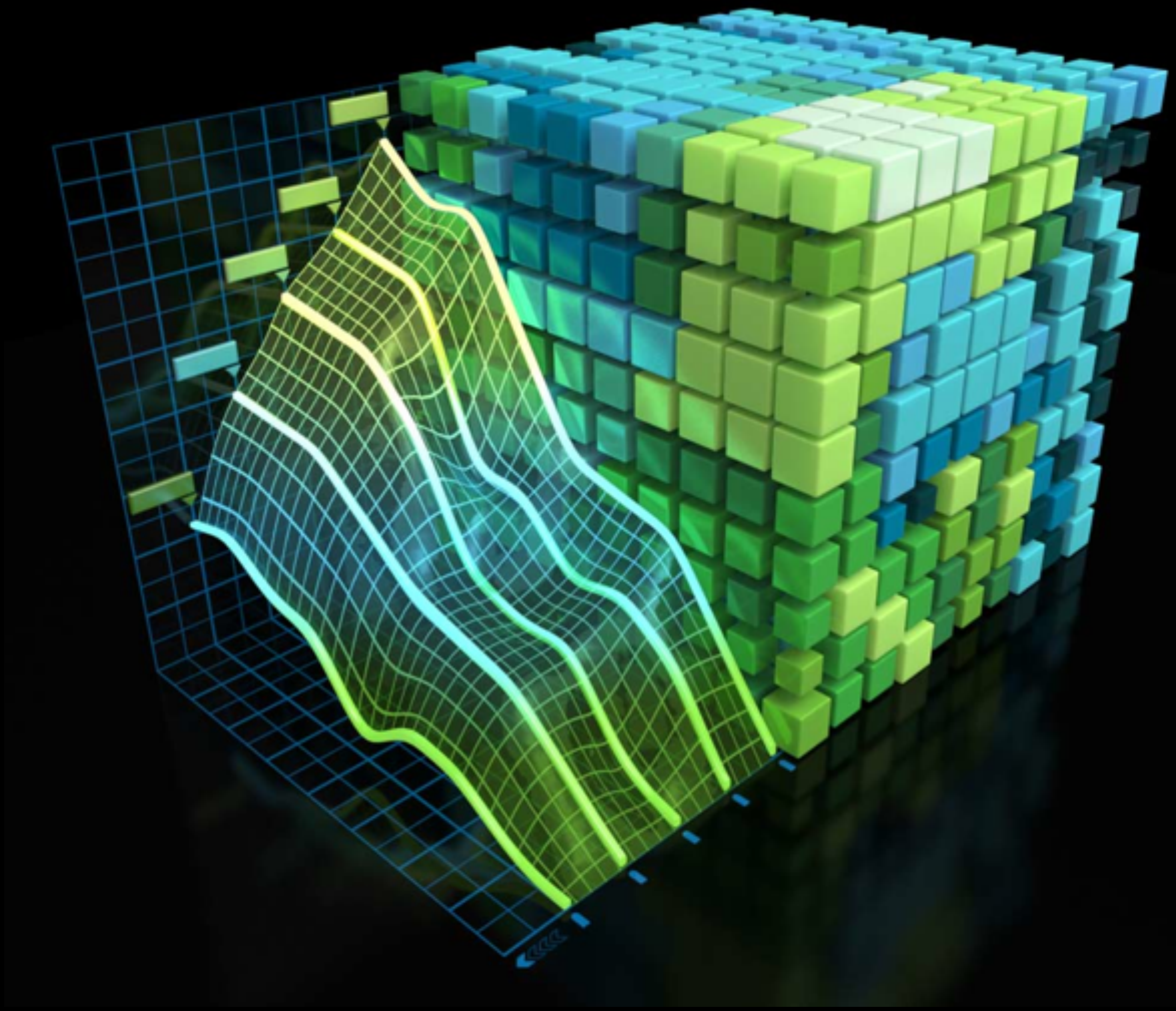


Few cores

Hundreds cores

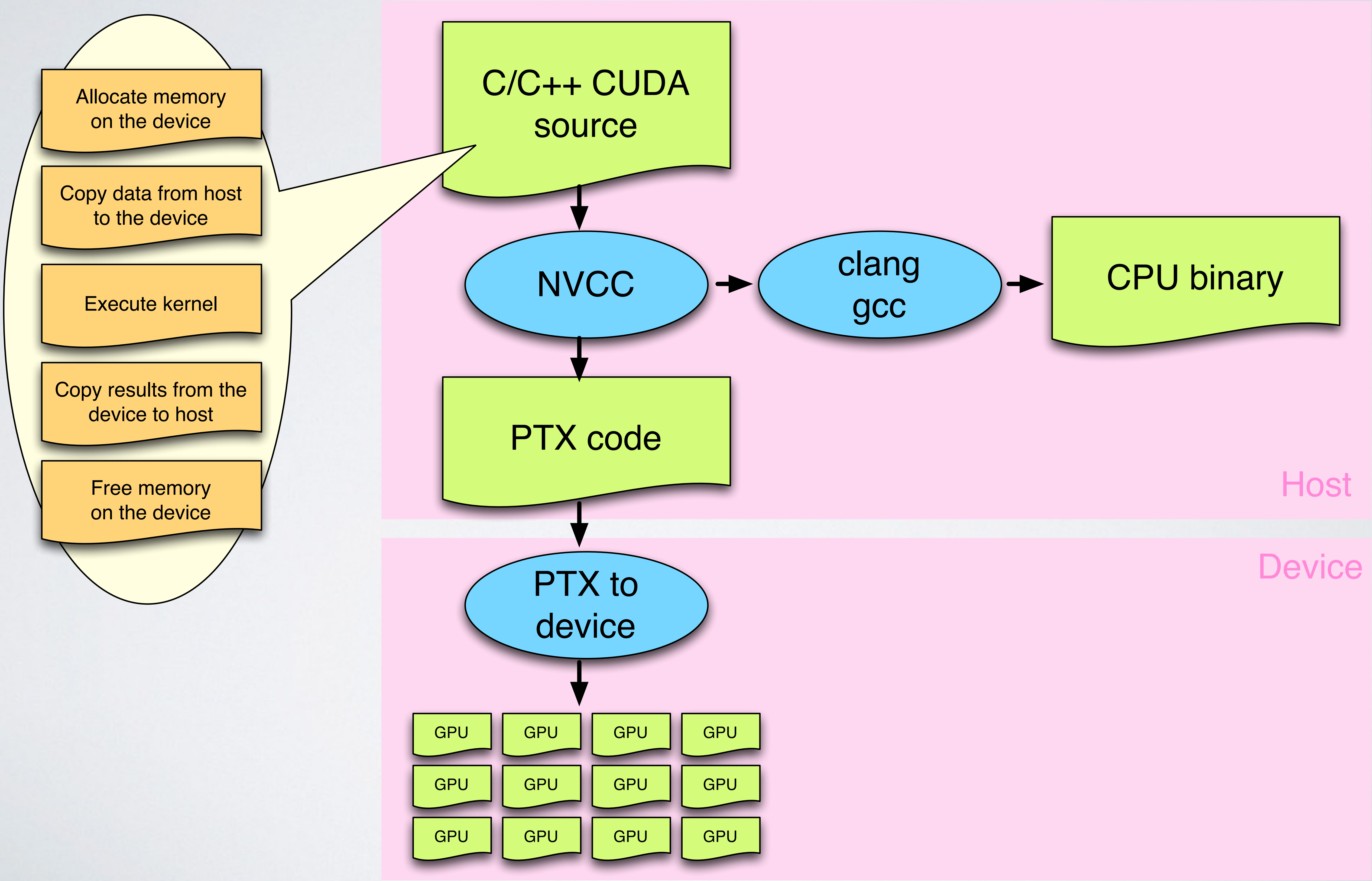
CUDA

"COMPUTE UNIFIED DEVICE ARCHITECTURE"



NVIDIA Corporation
ver 1.0: 2007
ver 6.0: Apr 2014

TYPICAL PROGRAM



Allocate memory on the device

Copy data from host to the device

Execute kernel

Copy results from the device to host

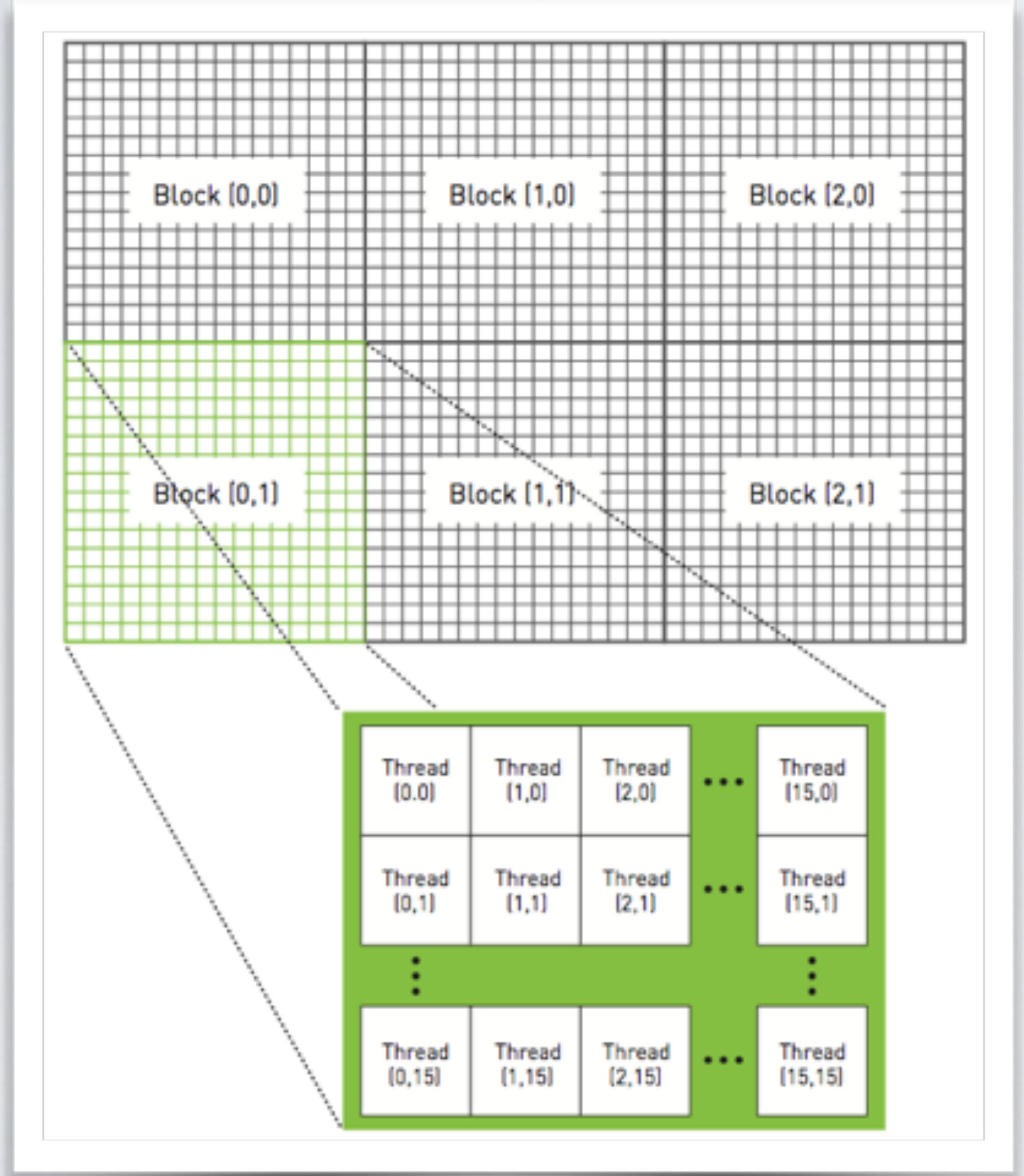
Free memory on the device

SIMPLE SAMPLE

```
#include <cuda.h>
#define N 10

__global__ void kernel(int *a, int *b, int *c) {
    int tid = blockIdx.x;
    if (tid < N) {
        c[tid] = a[tid] + b[tid];
    }
}

int main() {
    int a[N], b[N], c[N];
    int *dev_a, *dev_b, *dev_c;

    cudaMalloc((void**)&dev_a, N * sizeof(int));
    cudaMalloc((void**)&dev_b, N * sizeof(int));
    cudaMalloc((void**)&dev_c, N * sizeof(int));

    for (int i = 0; i < N; ++i) {
        a[i] = -i;
        b[i] = i * i;
    }
```

```
    cudaMemcpy(dev_a, a, N * sizeof(int),
          cudaMemcpyHostToDevice);
    cudaMemcpy(dev_b, b, N * sizeof(int),
          cudaMemcpyHostToDevice);

    kernel<<<N, 1>>>(dev_a, dev_b, dev_c);

    cudaMemcpy(c, dev_c, N * sizeof(int),
          cudaMemcpyDeviceToHost);

    for (int i = 0; i < N; ++i) {
        printf("%d + %d = %d\n", a[i], b[i], c[i]);
    }

    cudaFree(dev_a);
    cudaFree(dev_b);
    cudaFree(dev_c);

    return 0;
}
```

BLOCKS AND THREADS



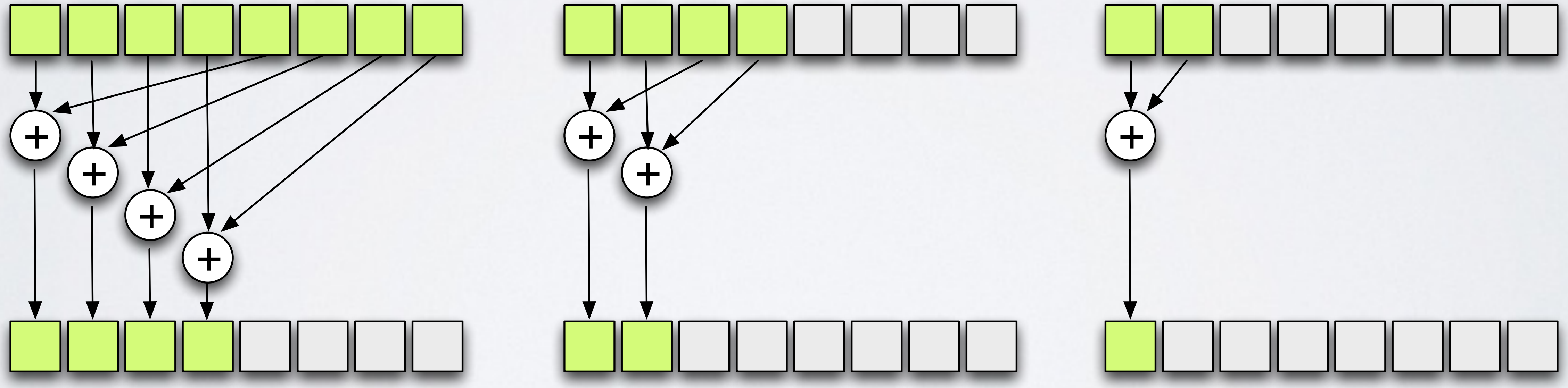
```
dim3 blocks(256, 256);
dim3 threads(16, 16);
kernel<<<block, threads>>>(data);
```

DEMO: MANDELBROT SET

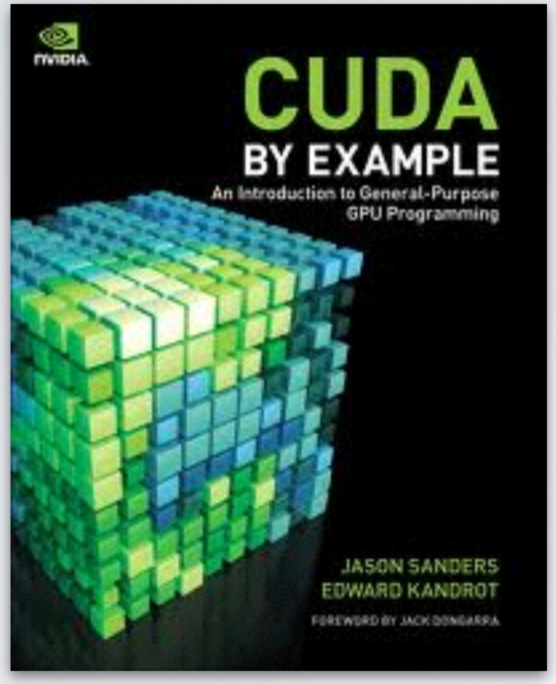
- Complex numbers
- Recurrent computation: $z_{n+1} = z_n^2 + c$
- c is the point of interest
- Start with $z_0 = 0+0i$
- c belongs to the set iff the absolute value of z_n remains bounded
- In practice: few hundred iterations
- If absolute value >= 2 then z_n will not be bounded

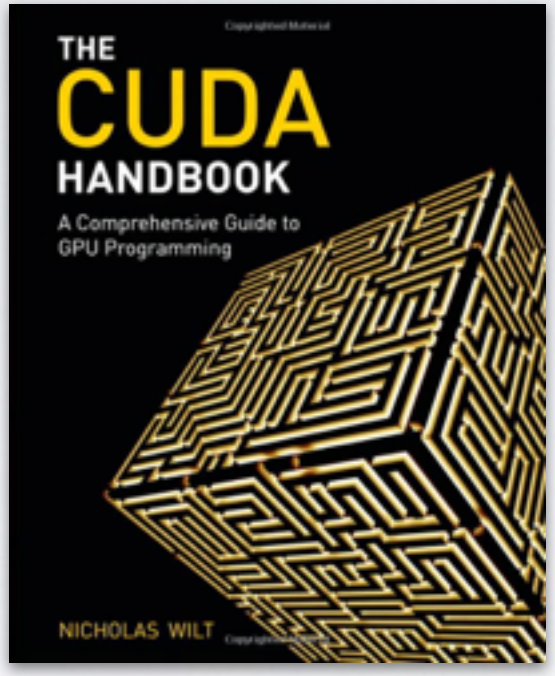
THINK PARALLEL

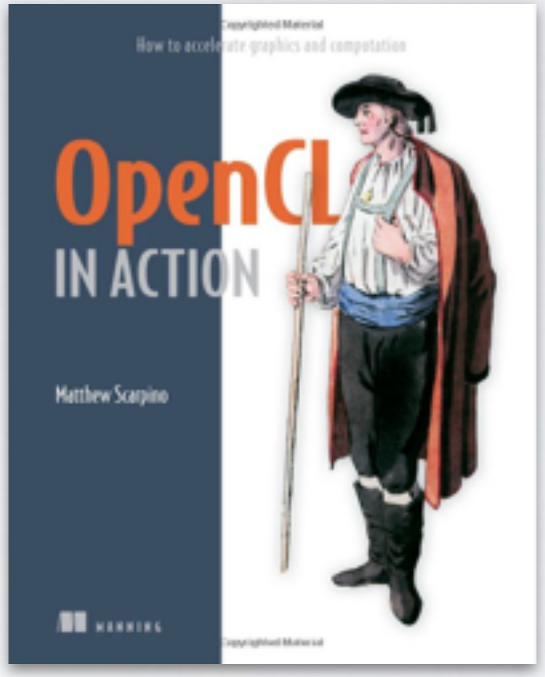
```
int sum_array(int[] a, int N) {
     int sum = 0;
     for (int i = 0; i < N; ++i) {
          sum += a[i];
     }
     return sum;
}
```

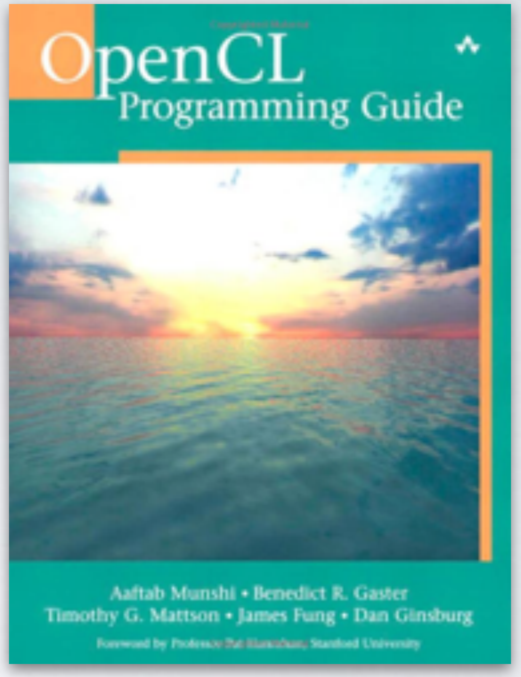


HUNGRY FOR MORE ?


NVIDIA
CUDA
BY EXAMPLE
An Introduction to General-Purpose GPU Programming
JASON SANDERS
EDWARD KANDROT
FOREWORD BY JACK DONGARRA






OpenCL
Programming Guide
Aaftab Munshi • Benedict R. Gaster
Timothy G. Mattson • James Fung • Dan Ginsburg
Foreword by Professor Pat Hanrahan, Stanford University

Introduction to Parallel Programming With CUDA – Udacity
https www.udacity.com/course/cs344
Reader
Plex
Calibre
Introduction to Parallel Programming With CUDA – Udacity
UDACITY
Course Catalog
Sign In
Sign Up
View Trailer

Intro to Parallel Programming

Using CUDA to Harness the Power of GPUs

https://github.com/tvarkoj/project-mandel