Lec 1 Hello CUDA

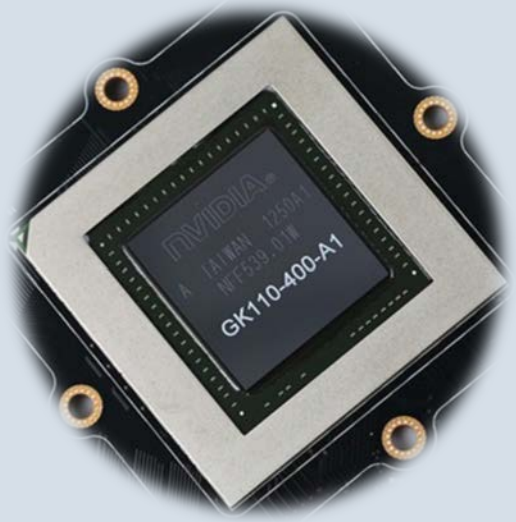
Dong Li, Tonghua Su
School of Software
Harbin Institute of Technology

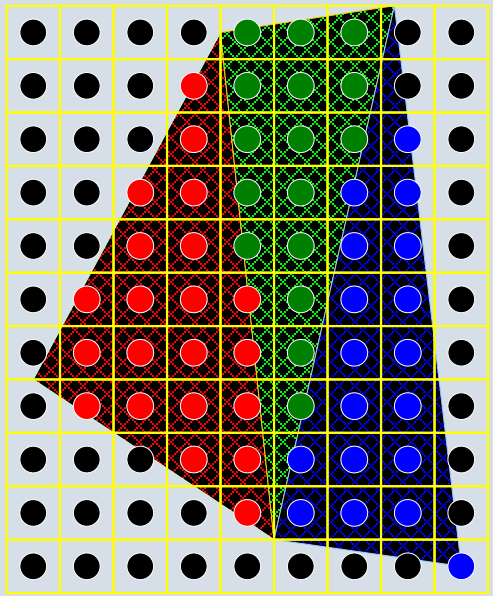
Outline

1. What is GPU?
2. NVIDIA GPU Architecture (briefly)
3. Hello CUDA
4. Amdahl's Law
5. Lab 1

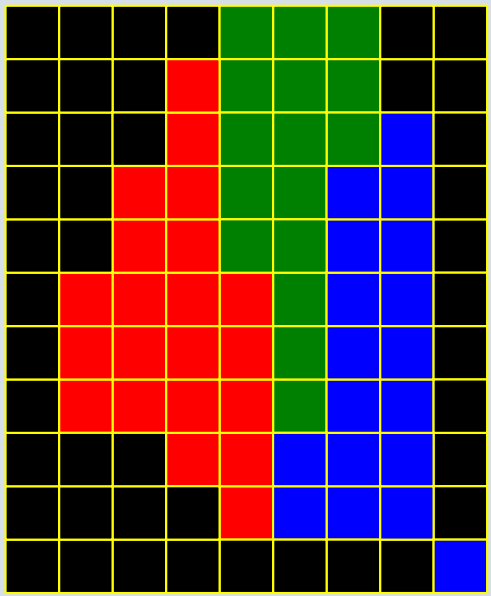
What is GPU

- Specialized electronic circuit
  - ✓ Accelerate the building of images intended for display

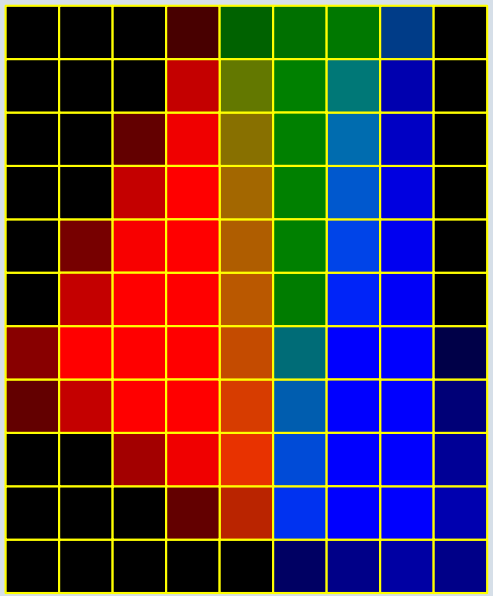




Triangle Geometry



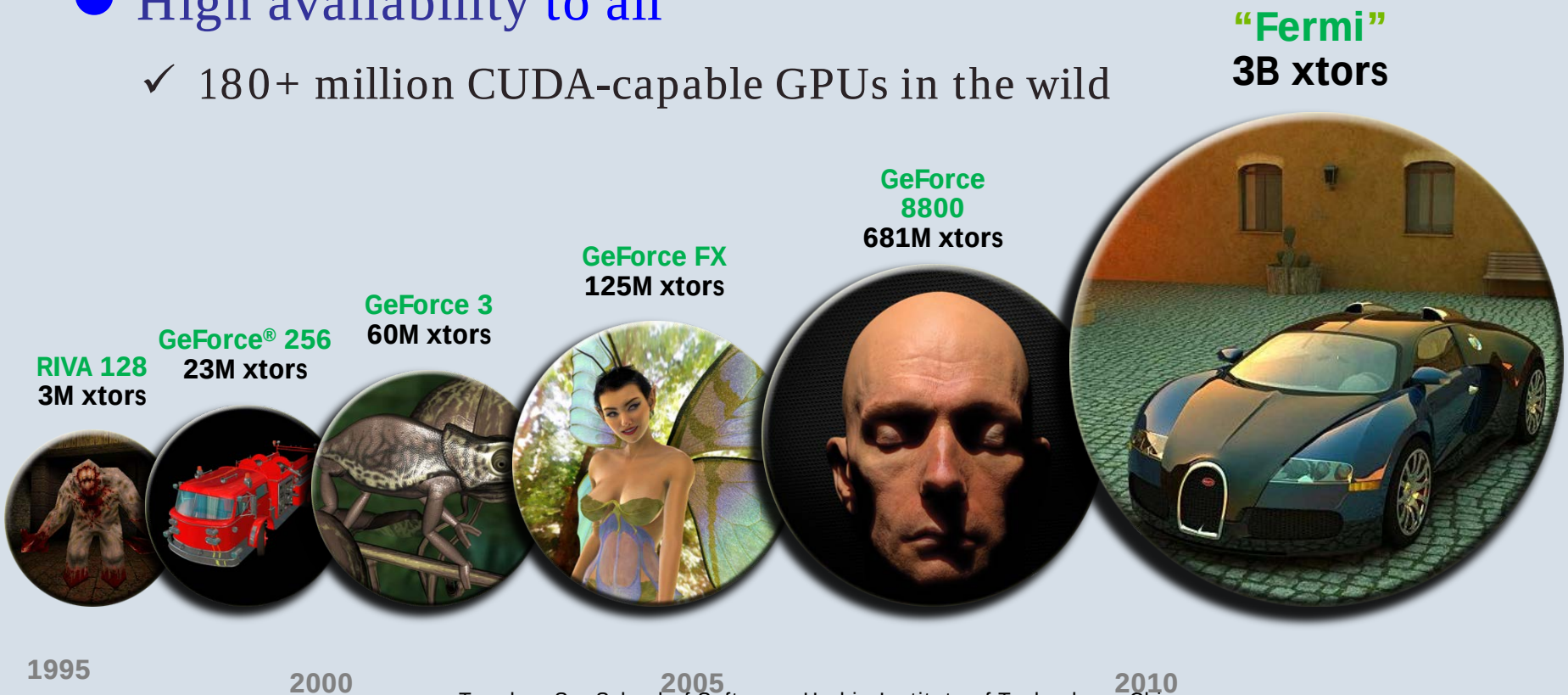
Aliased



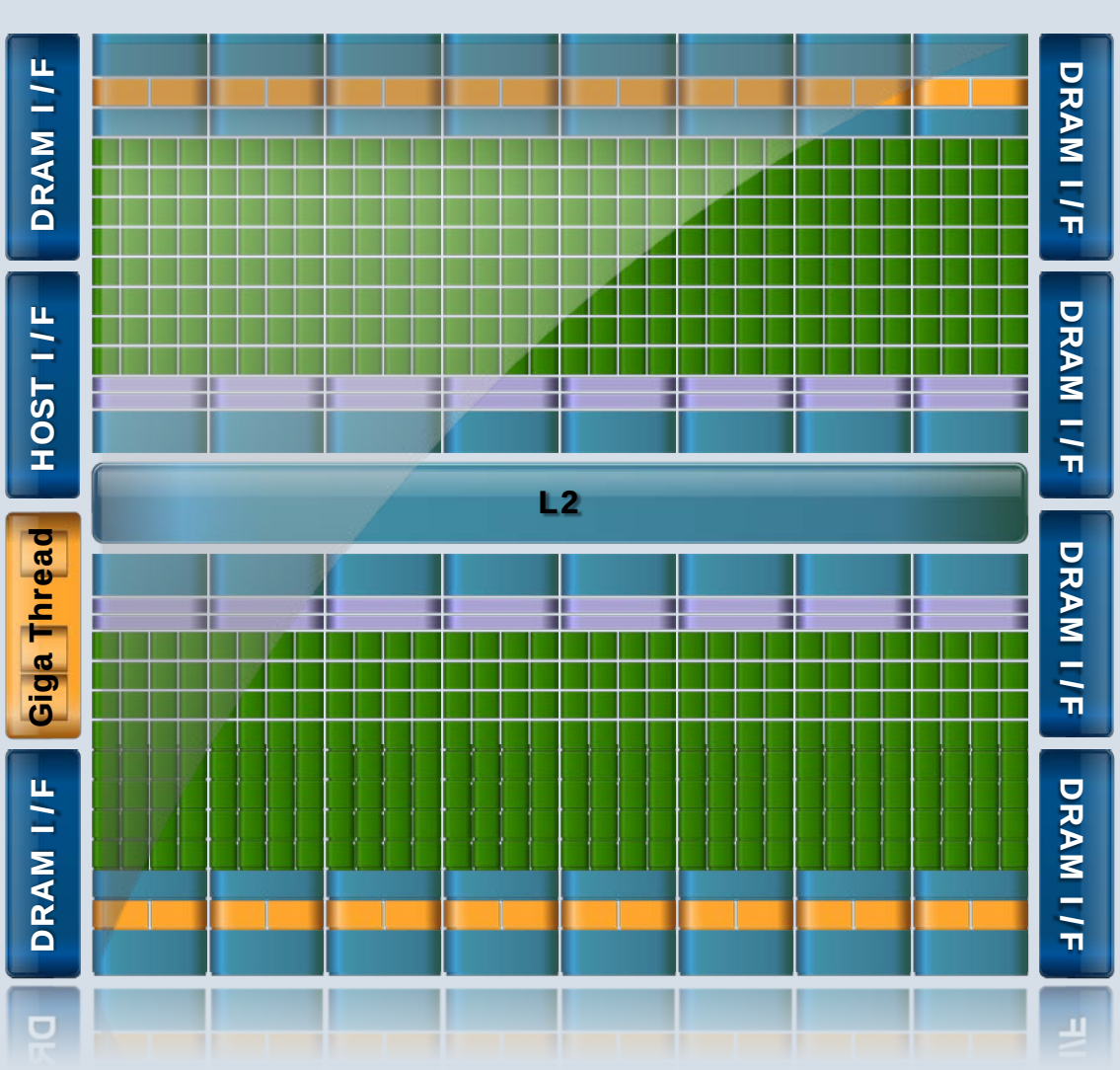
Anti-Aliased

What is GPU

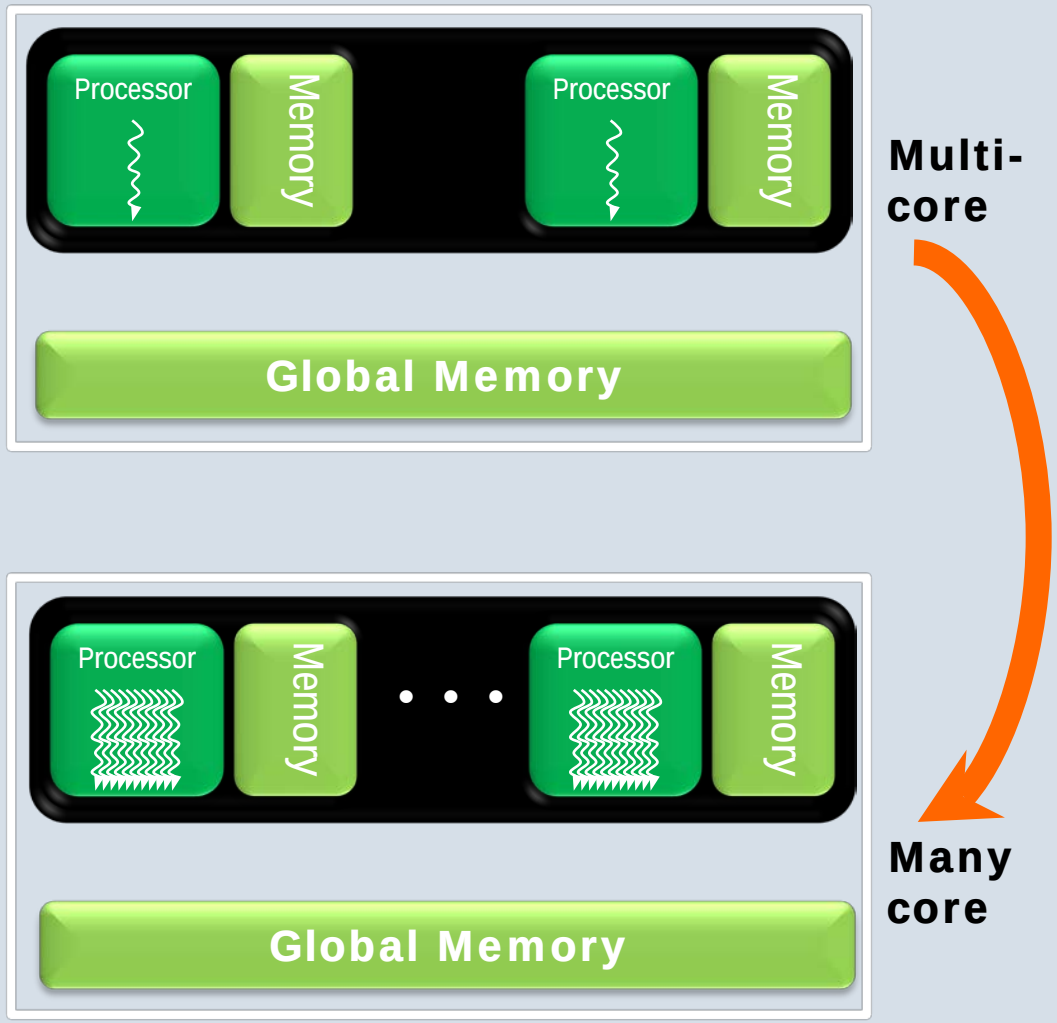
- High throughput computation
  - ✓ GeForce GTX 280: 933 GFLOP/s
- High bandwidth memory
  - ✓ GeForce GTX 280: 140 GB/s
- High availability to all
  - ✓ 180+ million CUDA-capable GPUs in the wild



NVIDIA GPU Architecture

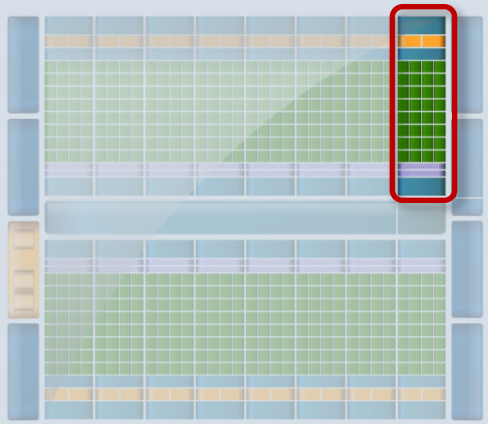


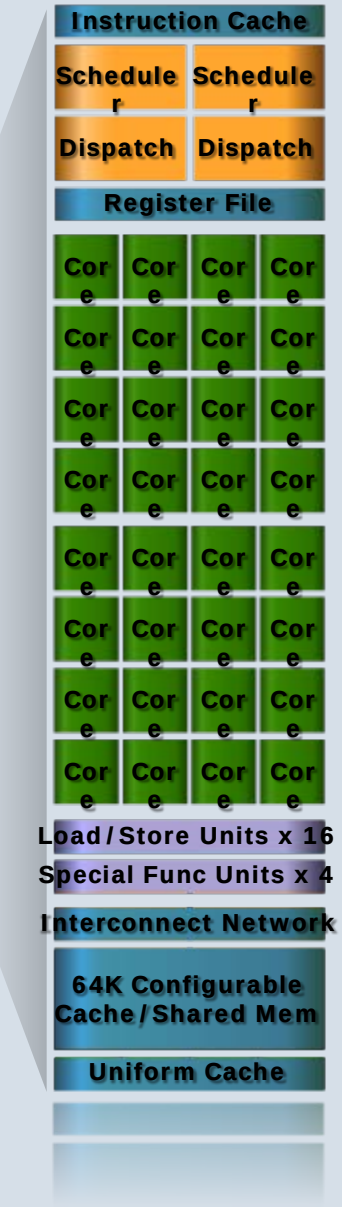
Fermi GF100



Streaming Multi-processor (SM)

- 32 CUDA Cores per SM (512 total)
- 8x peak FP64 performance
  - ✓ 50% of peak FP32 performance
- Direct load/store to memory
  - ✓ Usual linear sequence of bytes
  - ✓ High bandwidth (Hundreds GB/sec)
- 64KB of fast, on-chip RAM
  - ✓ Software or hardware-managed
  - ✓ Shared amongst CUDA cores
  - ✓ Enables thread communication



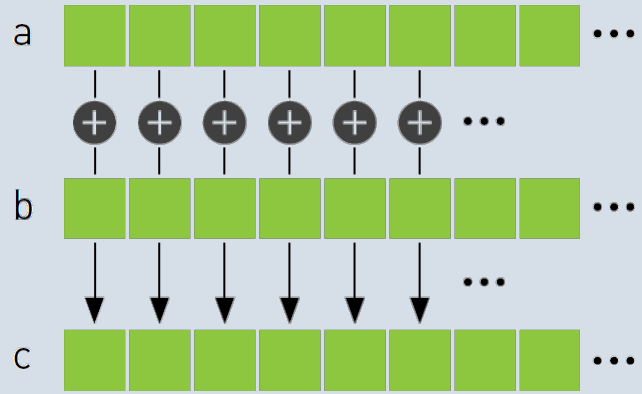


Lab 1.1 Device Query

- 查询你机器上GPU设备的参数
  - ✓ 新建.cu文件
  - ✓ 调用cudaGetDeviceCount()得到GPU设备的数量
  - ✓ 调用cudaGetDeviceProperties()函数得到GPU设备的属性结构体
  - ✓ 解释关键属性的含义，至少包括设备名称、计算能力为多少、设备可用全局内存、每线程块最大线程数、设备可用全局内存容量、每线程块可用共享内存容量、每线程块可用寄存器数量、每线程块最大线程数、每个处理器簇最大驻留线程数、设备中的处理器簇数量等
  - ✓ 可参考WILT 3.2节

Hello CUDA

- Vector Sum



✓ Serial code

```
for (i=0;i<128;i++)
{
        c[i] = a[i] + b[i];
}
```

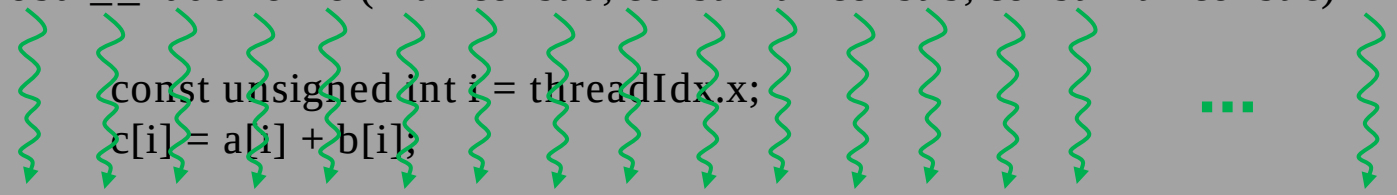
✓ Translate into CUDA threads

```
__global__ void addKernel(int * const a, const int * const b, const int * const c)
{
        c[i] = a[i] + b[i];
}
```

Hello CUDA

- Vector Sum
  - ✓ Identify thread id

```
__global__ addKernel(int * const a, const int * const b, const int * const c)
{
        const unsigned int i = threadIdx.x;
        c[i] = a[i] + b[i];
}
```

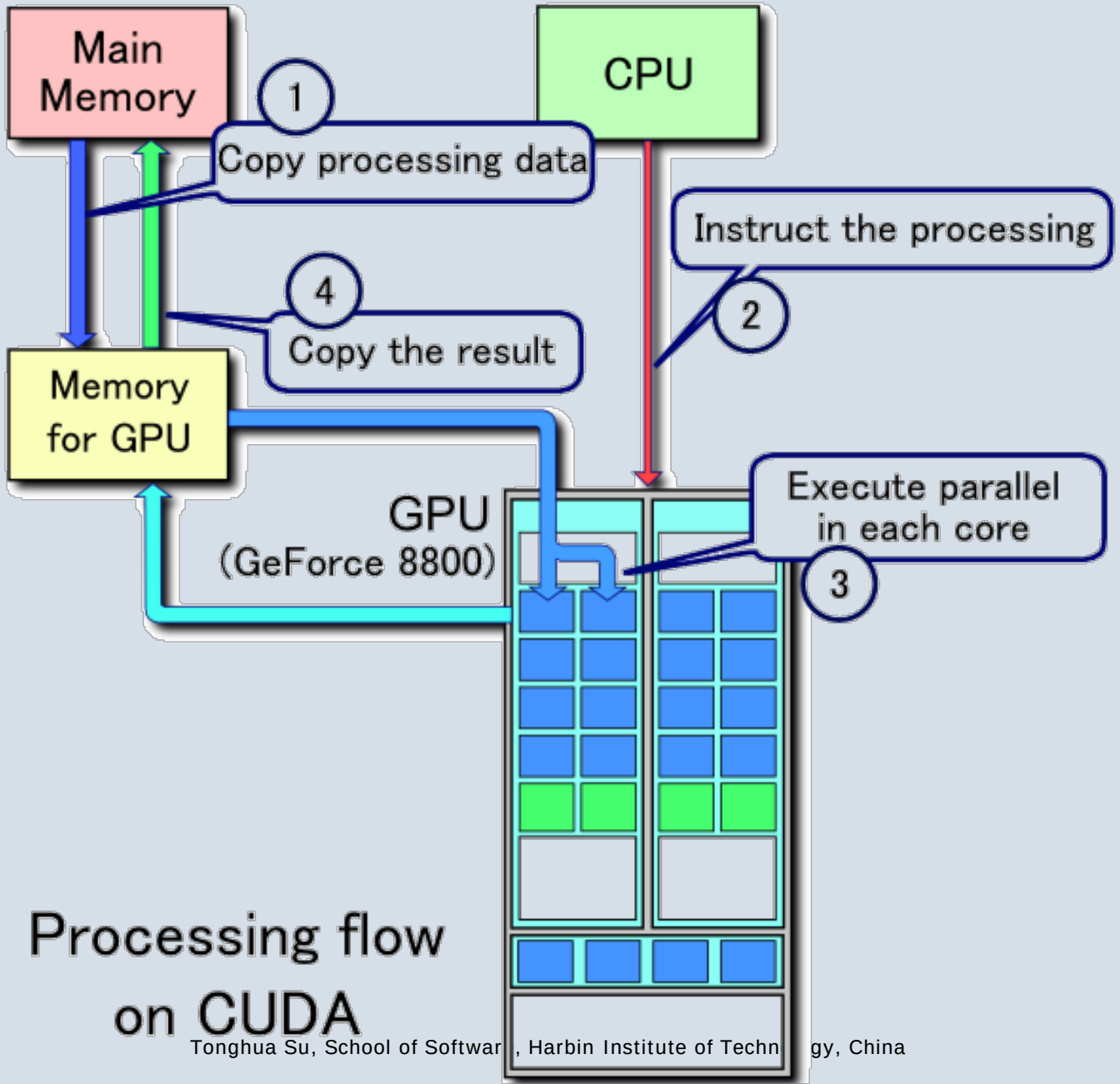


  - ✓ Invoke CUDA kernel
    - kernel_function<<<num_blocks, num_threads>>>(param1, param2, .)
    - E.g. addKernel<<< 1, 128 >>>(a, b, c);

Hello CUDA

- Demo 1

Hello CUDA


Main Memory
CPU
1
Copy processing data
Instruct the processing
2
4
Copy the result
Memory for GPU
GPU
(GeForce 8800)
Execute parallel in each core
3
Processing flow on CUDA

Hello CUDA

```
__global__ void addKernel(int *c, const int *a, const int *b)
{
        int i = threadIdx.x;
        c[i] = a[i] + b[i];
}
```

```
void main(){
        ......
        int *dev_a,*dev_b,*dev_c;
        // Allocate GPU buffers for three vectors (two input, one output)    .
        cudaMalloc((void**)&dev_c, 128* sizeof(int));
        cudaMalloc((void**)&dev_a, 128* sizeof(int));
        cudaMalloc((void**)&dev_b, 128* sizeof(int));

        // Copy input vectors from host memory to GPU buffers.
        cudaMemcpy(dev_a, a, 128* sizeof(int), cudaMemcpyHostToDevice);
        cudaMemcpy(dev_b, b, 128* sizeof(int), cudaMemcpyHostToDevice);

        // Launch a kernel on the GPU with one thread for each element.
        addKernel<<<1, 128>>>(dev_c, dev_a, dev_b);

        // Copy output vector from GPU buffer to host memory.
        cudaMemcpy(c, dev_c, 128* sizeof(int), cudaMemcpyDeviceToHost);

        cudaFree(dev_c);
        cudaFree(dev_a);
        cudaFree(dev_b);
}
```

Lab 1.2 Error Handling

- 尝试多种查看错误的方式
  - ✓ 在内核函数内printf信息
  - ✓ 组合调用cudaGetLastError()和cudaGetErrorString()，返回出错字符串
  - ✓ 宏
  - ✓ 在内核启动时，使用非法参数，检验是否成功处理
  - ✓ 可参考WILT 附录A.3节

Amdahl's Law

- Gene Amdahl in 1967:

$$Speedup = \frac{1}{r_s + \frac{r_p}{N}}$$

where $r_s + r_p = 1$ and r_s represents the ratio of the sequential portion.

✓ Consider: 30% portion of time with 100X speedup through paralleling vs 99% portion with 100X

Amdahl, Gene (1967). "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". *AFIPS Conference Proceedings* (30): 483–485.