CUDA/GPU 编程模型

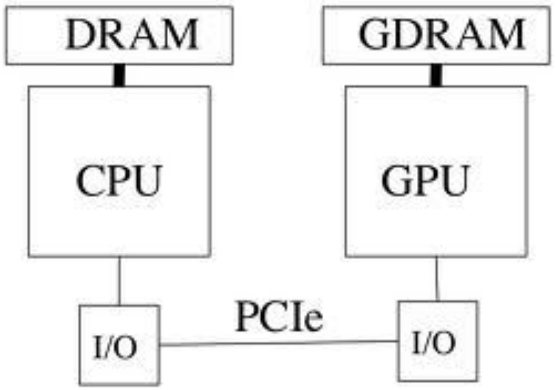
周　斌 @NVIDIA & USTC 2014年3月

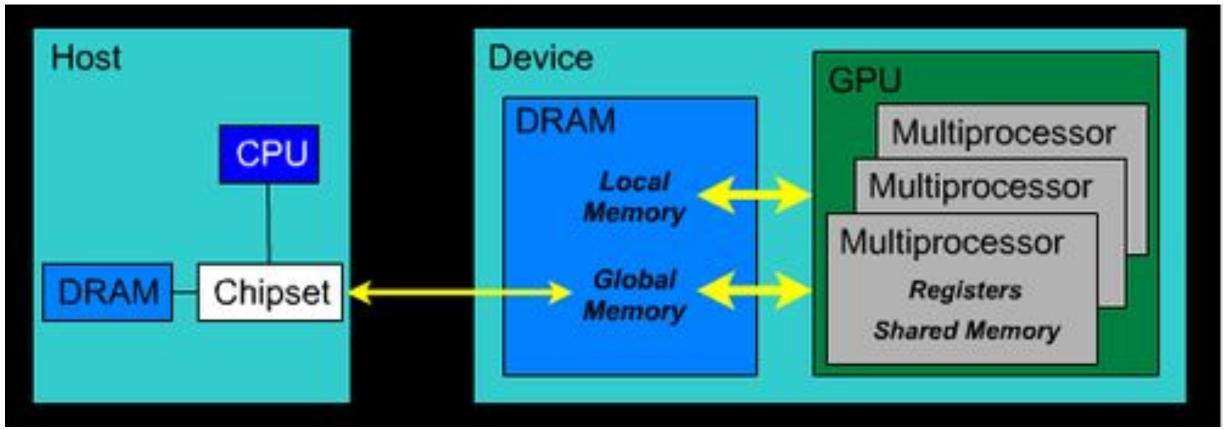
内容

- CPU和GPU互动模式
- GPU线程组织模型（不停强化）
- GPU存储模型
- 基本的编程问题

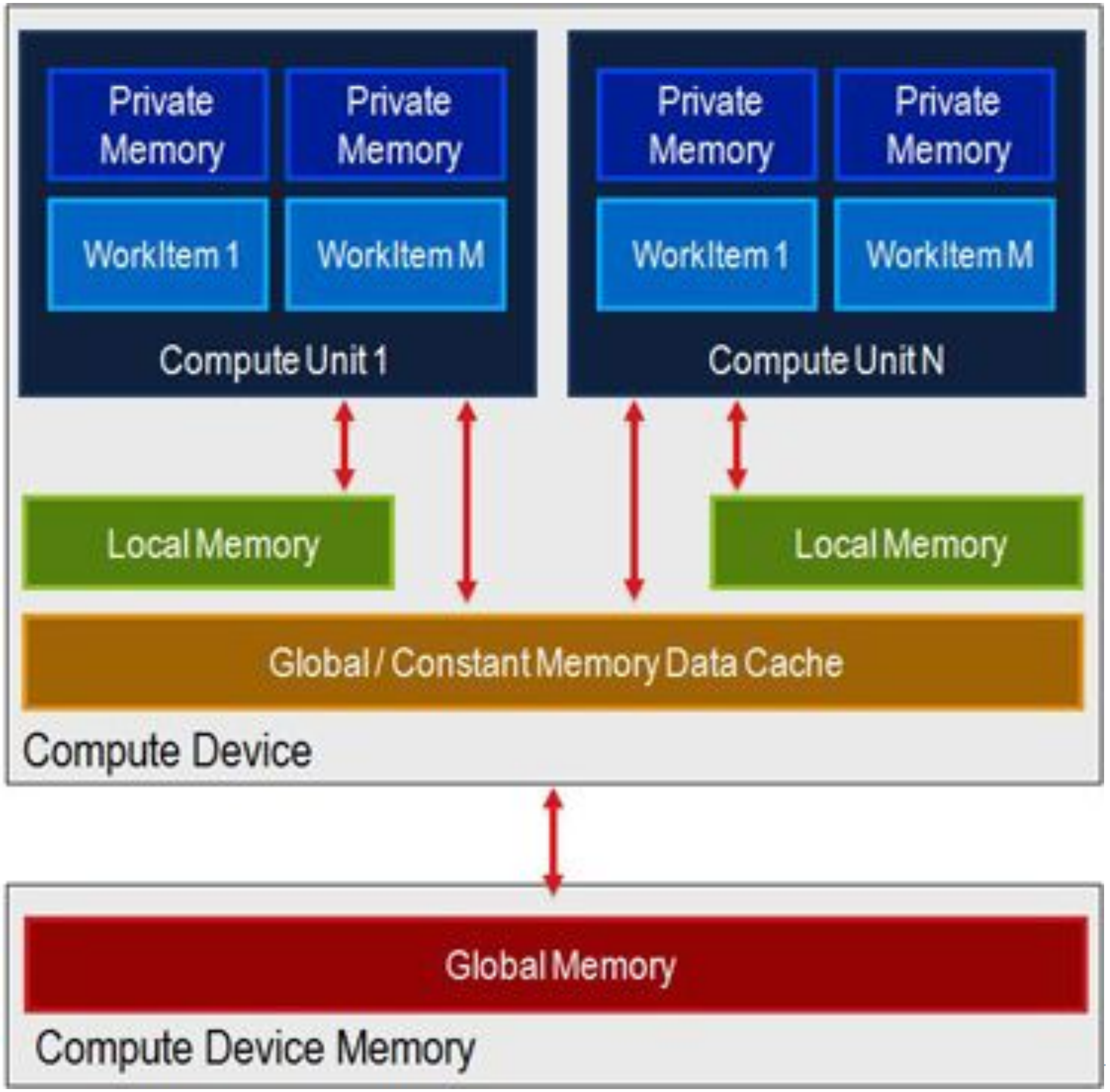
CPU-GPU交互

- 各自的物理内存空间
- 通过PCIE总线互连(8GB/s～16GB/s)
- 交互开销较大





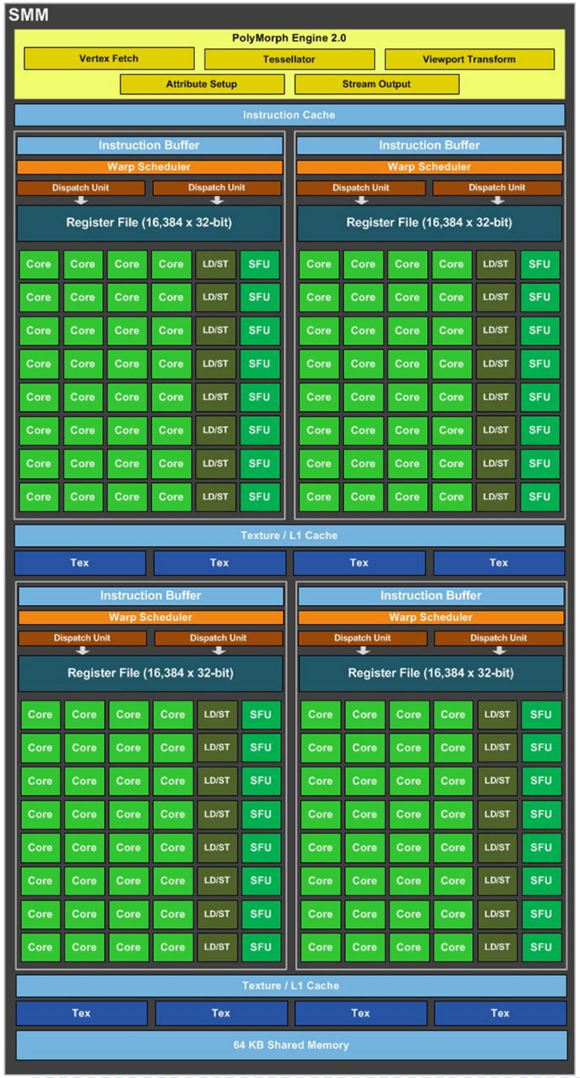
GPU存储器层次架构（硬件）

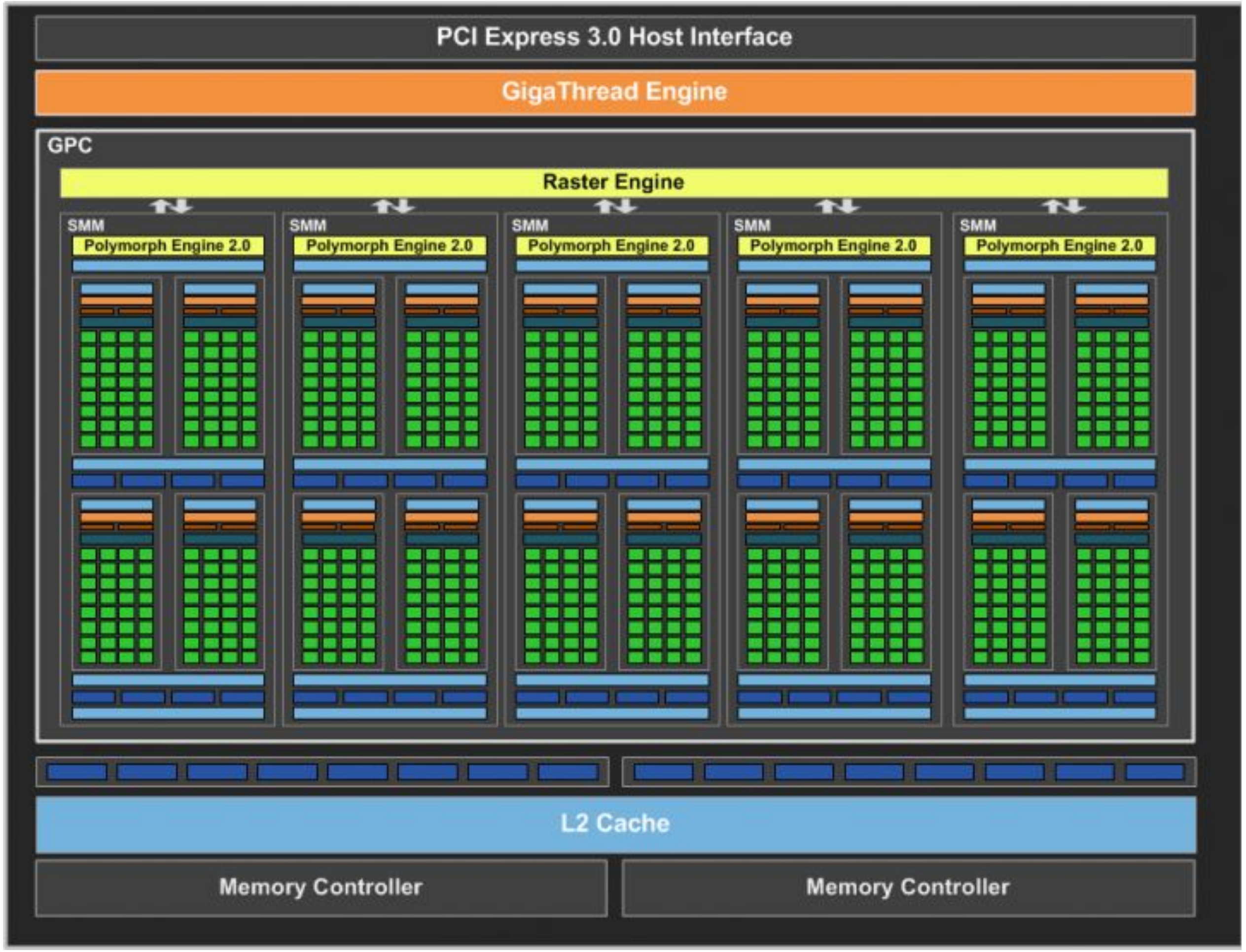


访存速度

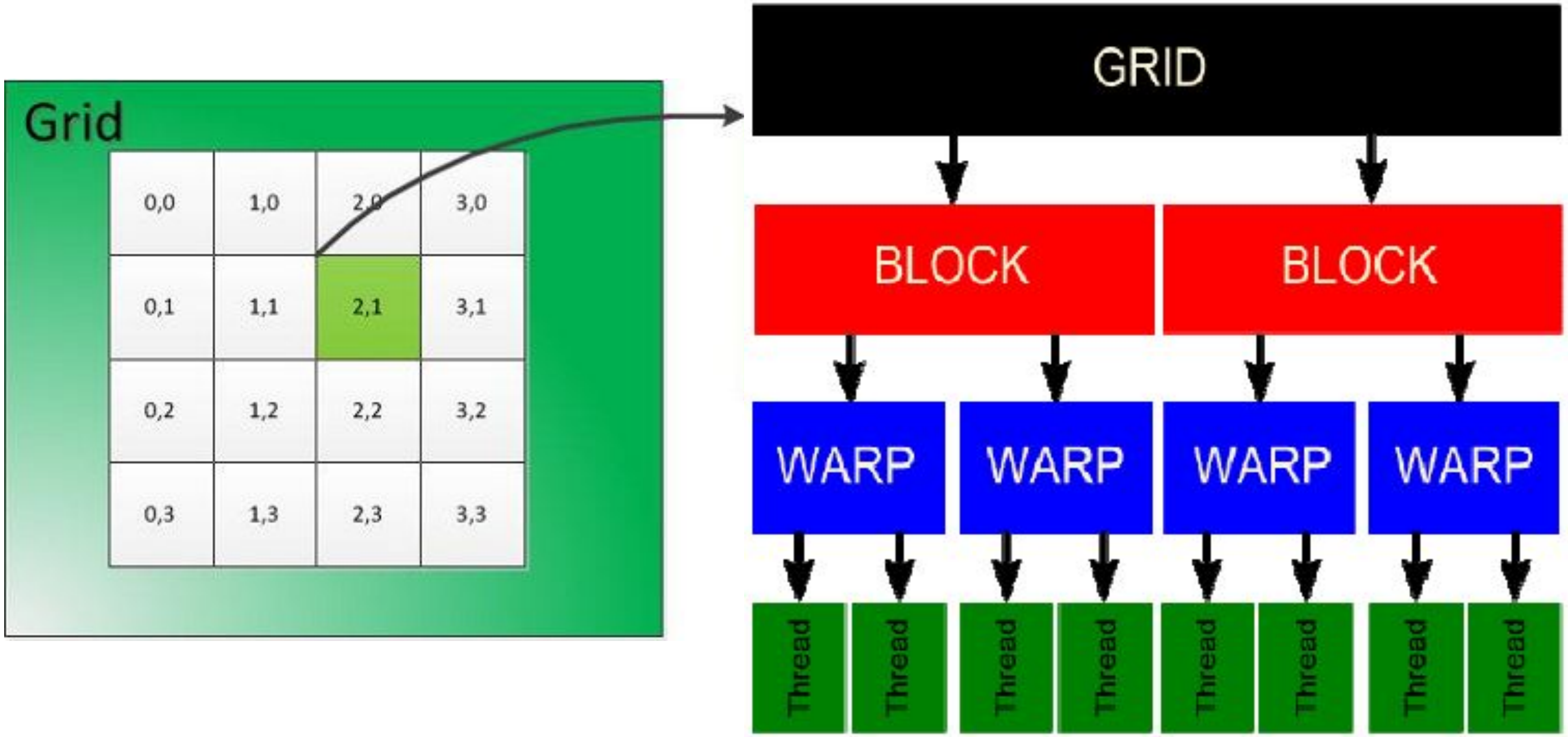
- Register – dedicated HW - single cycle
- Shared Memory – dedicated HW - single cycle
- Local Memory – DRAM, no cache - *slow*
- Global Memory – DRAM, no cache - *slow*
- Constant Memory – DRAM, cached, 1…10s…100s of cycles, depending on cache locality
- Texture Memory – DRAM, cached, 1…10s…100s of cycles, depending on cache locality
- Instruction Memory (invisible) – DRAM, cached

GPU架构回顾


SMM
PolyMorph Engine 2.0
Vertex Fetch
Tessellator
Viewport Transform
Attribute Setup
Stream Output
Instruction Cache
Instruction Buffer
Warp Scheduler
Dispatch Unit
Register File (16,384 x 32-bit)
Core
LD/ST
SFU
Texture / L1 Cache
Tex
64 KB Shared Memory


PCI Express 3.0 Host Interface
GigaThread Engine
GPC
Raster Engine
SMM
Polymorph Engine 2.0
L2 Cache
Memory Controller
Memory Controller

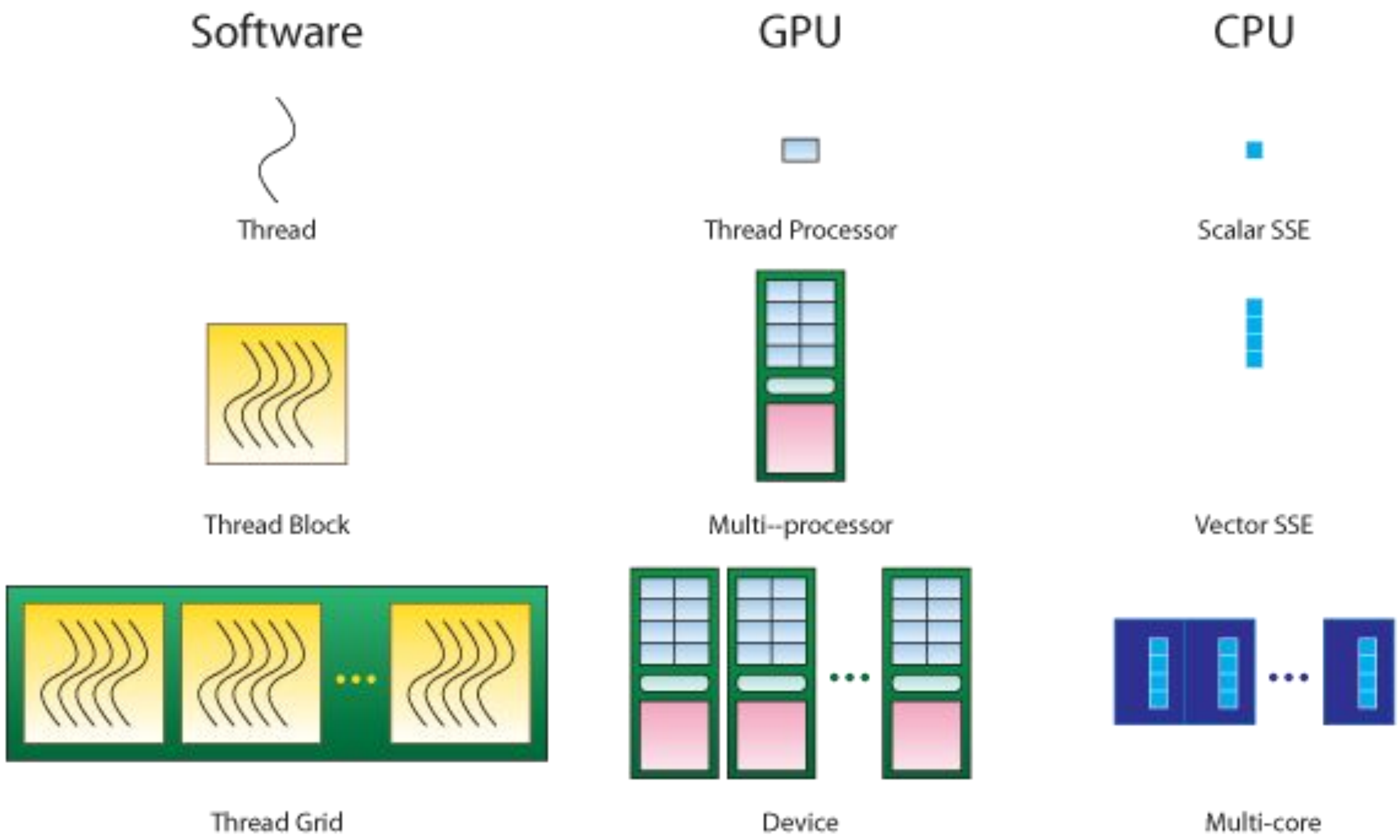
GPU线程组织模型


Grid
0,0
1,0
2,0
3,0
0,1
1,1
2,1
3,1
0,2
1,2
2,2
3,2
0,3
1,3
2,3
3,3
GRID
BLOCK
BLOCK
WARP
WARP
WARP
WARP
Thread
Thread
Thread
Thread
Thread
Thread
Thread
Thread

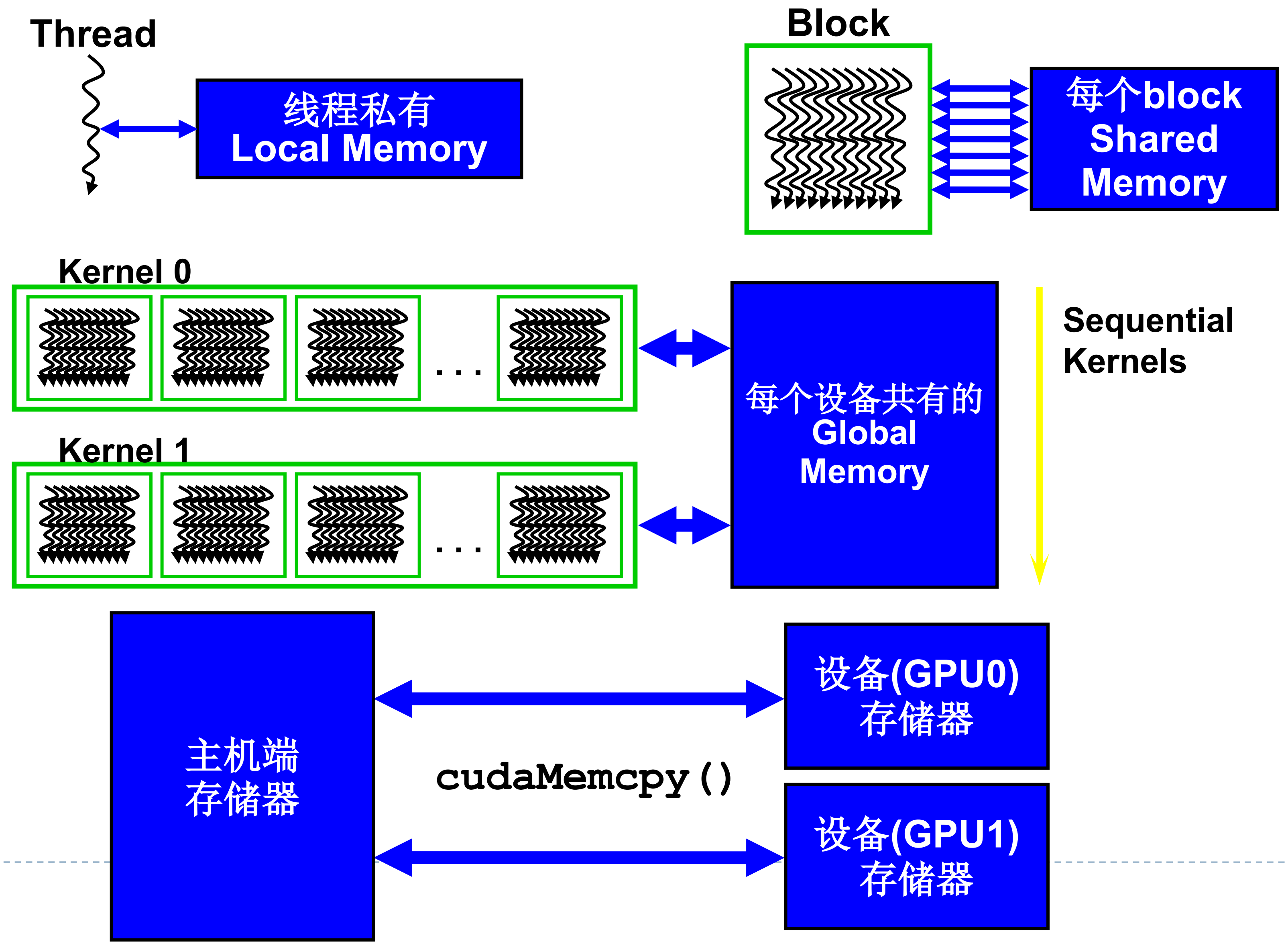
线程组织架构说明

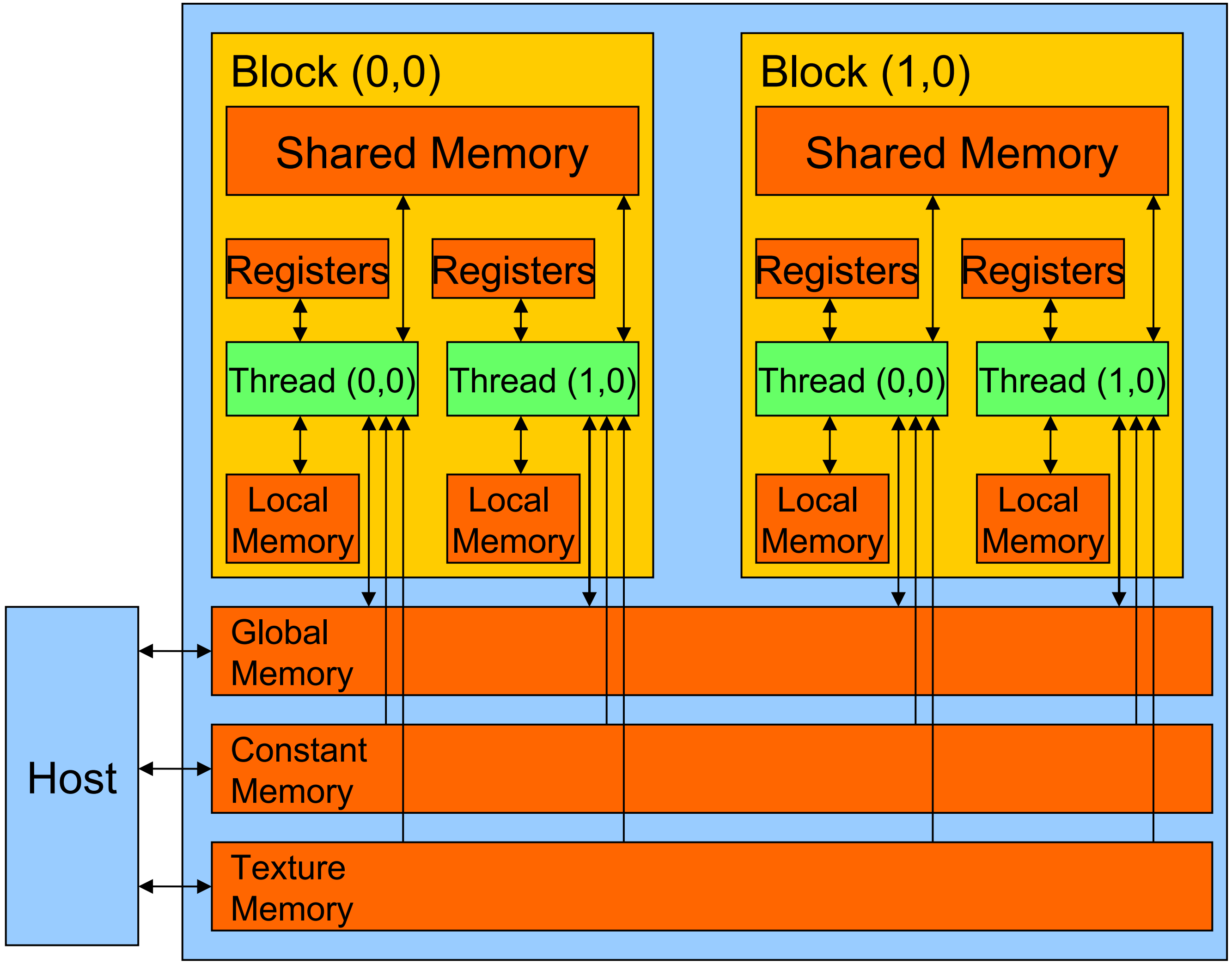
- 一个Kernel具有大量线程
- 线程被划分成线程块‘blocks’
  - 一个block内部的线程共享‘Shared Memory’
  - 可以同步‘_syncthreads()’
- Kernel启动一个‘grid’，包含若干线程块
  - 用户设定
- 线程和线程块具有唯一的标识

GPU线程映射关系


Software
GPU
CPU
Thread
Thread Processor
Scalar SSE
Thread Block
Multi--processor
Vector SSE
Thread Grid
Device
Multi-core

GPU内存和线程等关系




Block (0,0)
Shared Memory
Registers
Registers
Thread (0,0)
Thread (1,0)
Local Memory
Local Memory
Block (1,0)
Shared Memory
Registers
Registers
Thread (0,0)
Thread (1,0)
Local Memory
Local Memory
Host
Global Memory
Constant Memory
Texture Memory

编程模型

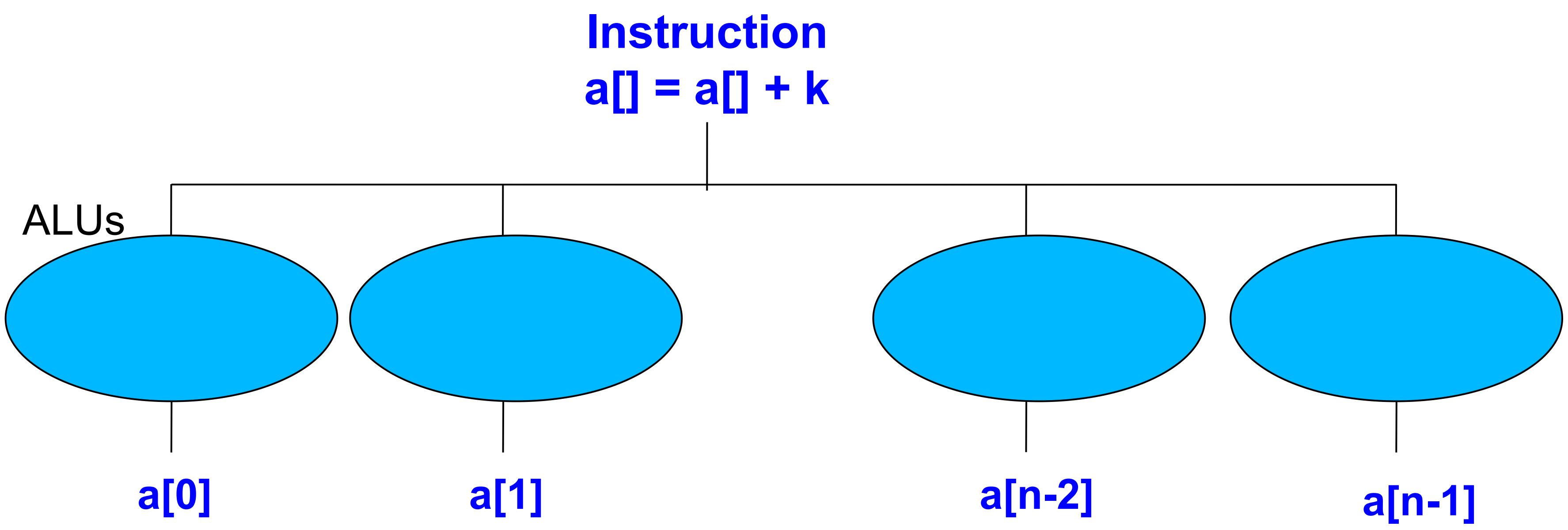
常规意义的GPU用于处理图形图像

操作于像素，每个像素的操作都类似

可以应用SIMD (single instruction multiple data)

SIMD (Single Instruction Multiple Data)

也可以认为是数据并行分割



Single Instruction Multiple Thread (SIMT)

GPU版本的 SIMD

大量线程模型获得高度并行

线程切换获得延迟掩藏

多个线程执行相同指令流

GPU上大量线程承载和调度

CUDA编程模式：Extended C

- Declspecs
  - global, device, shared, local, constant
- 关键词
  - threadIdx, blockIdx
- Intrinsics
  - __syncthreads
- 运行期API
  - Memory, symbol, execution management
- 函数调用

```
__device__ float filter[N];

__global__ void convolve (float *image)  {

  __shared__ float region[M];
  ...

  region[threadIdx] = image[i];

  __syncthreads()
  ...

  image[j] = result;
}

// Allocate GPU memory
void *myimage = cudaMalloc(bytes)

// 100 blocks, 10 threads per block
convolve<<<100, 10>>> (myimage);
```

CUDA 函数声明

	执行位置	调用位置
__device__ float DeviceFunc()	device	device
__global__ void KernelFunc()	device	host
__host__ float HostFunc()	host	host

- __global__ 定义一个 kernel 函数
  - 入口函数，CPU上调用，GPU上执行
  - 必须返回void
- __device__ and __host__ 可以同时使用