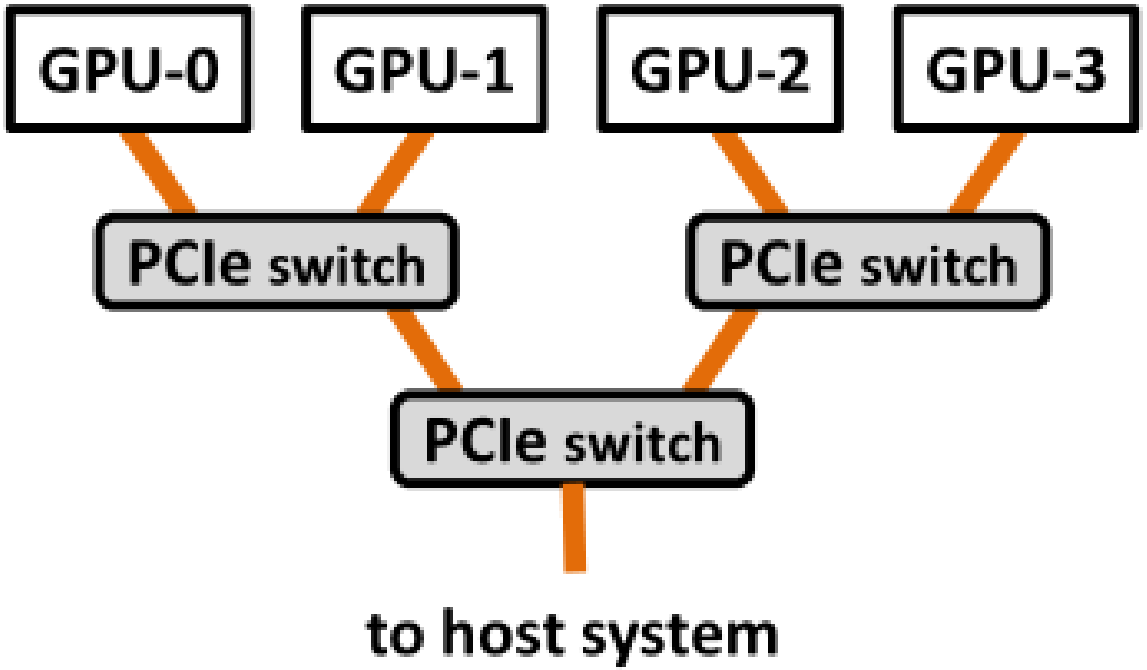


GPU并行计算与CUDA编程 第6课

第一版 讲师 罗韵（WeChat：LaurenLuoYun）

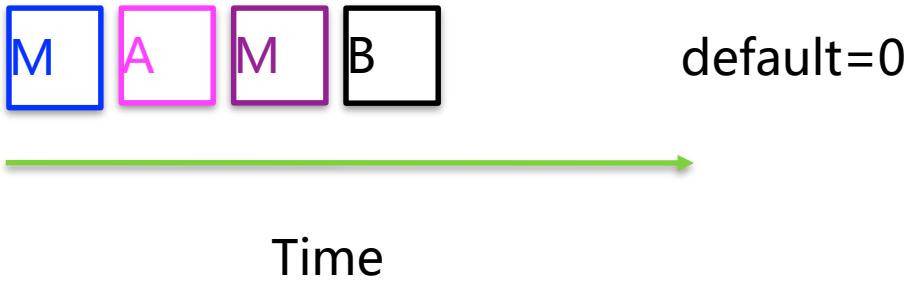
本周介绍内容

- 1. CUDA流Streams
- 2. 多GPU编程
- 3. 纹理内存与纹理操作
- 4. CPU/GPU协同

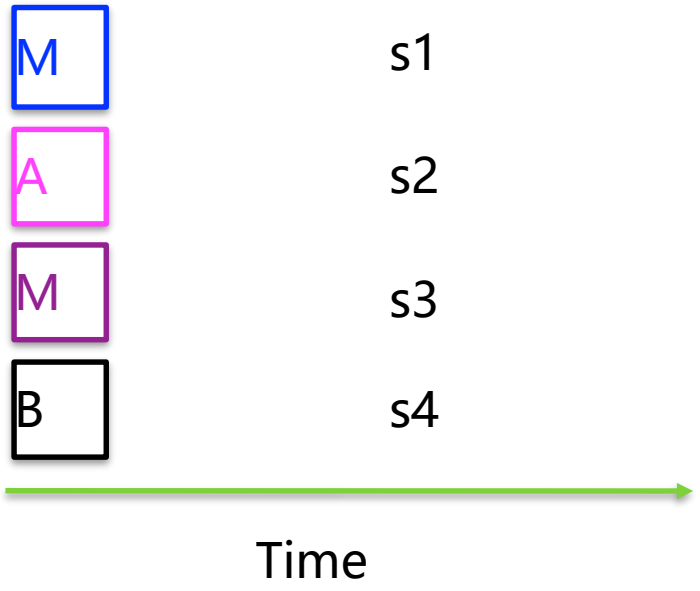
1. 流Streams

- 流：一系列将在GPU上按顺序执行的操作

```
cudaMemcpyAsync(...);
A<<<...>>>(...);
cudaMemcpyAsync(...);
B<<<...>>>(...);
```




```
cudaMemcpyAsync(...,s1);
A<<<...>>>(...,s2);
cudaMemcpyAsync(...,s3);
B<<<...>>>(...,s4);
```



- 定义流：cudaStream_t s1;
- 创建流：cudaStreamCreate(&s1);
- 销毁流：cudaStreamDestory(s1);

- 例子：

```
cudaStream_t s1, s2;
cudaStreamCreate(&s1); cudaStreamCreate(&s2);

cudaMemcpy(&d_arr, &h_arr, numbytes, cudaH2D);
A<<<1, 128>>>(d_arr);
cudaMemcpy(&h_arr, &d_arr, numbytes, cudaD2H);
```

1

```
cudaMemcpyAsync(&d_arr, &h_arr, numbytes, cudaH2D, s1);
A<<<1, 128, s1>>>(d_arr);
cudaMemcpyAsync(&h_arr, &d_arr, numbytes, cudaD2H, s1);
```

2

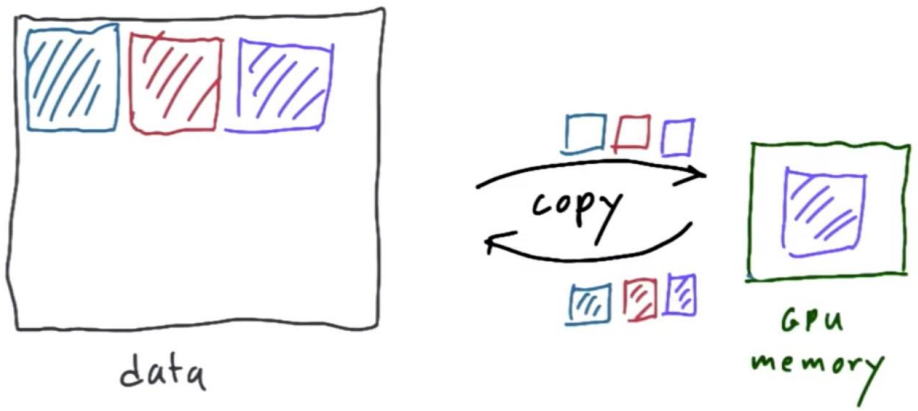
```
cudaMemcpyAsync(&d_arr1, &h_arr1, numbytes, cudaH2D, s1);
A<<<1, 128, s1>>>(d_arr1);
cudaMemcpyAsync(&h_arr1, &d_arr1, numbytes, cudaD2H, s1);
cudaMemcpyAsync(&d_arr2, &h_arr2, numbytes, cudaH2D, s2);
B<<<1, 192, s2>>>(d_arr2);
cudaMemcpyAsync(&h_arr2, &d_arr2, numbytes, cudaD2H, s2);
```

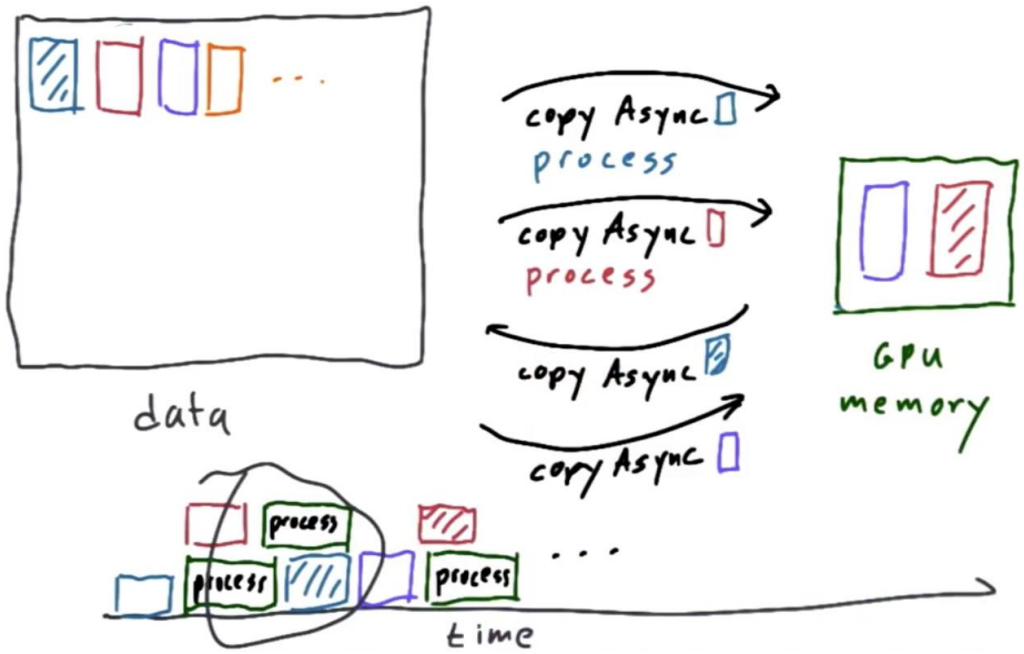
3

```
cudaMemcpyAsync(&d_arr1, &h_arr1, numbytes, cudaH2D, s1);
cudaMemcpyAsync(&d_arr2, &h_arr2, numbytes, cudaH2D, s2);
A<<<1, 128, s1>>>(d_arr1);
B<<<1, 192, s2>>>(d_arr2);
cudaMemcpyAsync(&h_arr1, &d_arr1, numbytes, cudaD2H, s1);
cudaMemcpyAsync(&h_arr2, &d_arr2, numbytes, cudaD2H, s2);
```

4

- 流的用处：


data
copy
GPU
memory


data
copy Async
process
copy Async
process
copy Async
copy Async
GPU
memory
process
process
process
time

2. 多GPU编程

- 统一地址：
- CPU和GPU分配使用统一的虚拟地址空间
  - 驱动/设备可以判断数据所在的地址
- GPU可以引用指针
  - 另一个GPU上的地址
  - Host上的地址

- 两个方面
  - Peer-to-peer(P2P) memcopies
  - 使用另一个GPU的地址
- cudaDeviceEnablePeerAccess(peer_device, 0)

  允许current GPU访问peer_device GPU
- cudaDeviceCanAccessPeer(&accessible, dev_X, dev_Y)

  检查是否dev_X可以访问dev_Y的内存返回0/1(第一个参数)

- 例子：

```
int gpu1 = 0;
int gpu2 = 1;
cudaSetDevice( gpu1 );
cudaMalloc( &d_A, num_bytes );
int accessible = 0;
cudaDeviceCanAccessPeer( &accessible, gpu2, gpu1 );
if( accessible )
{
cudaSetDevice( gpu2 );
cudaDeviceEnablePeerAccess( gpu1, 0 );
kernel<<<...>>>( d_A);
}
```

虽然内核在 Gpu2上执行，它可以访问在gpu1上分配的内存（通过PCIe）

- Peer-to-peer memcopy
- cudaMemcpyPeerAsync(void* dst_addr, intdst_dev, void* src_addr, intsrc_dev, size_tnum_bytes, cudaStream_tstream)

  两个设备之间拷贝字节

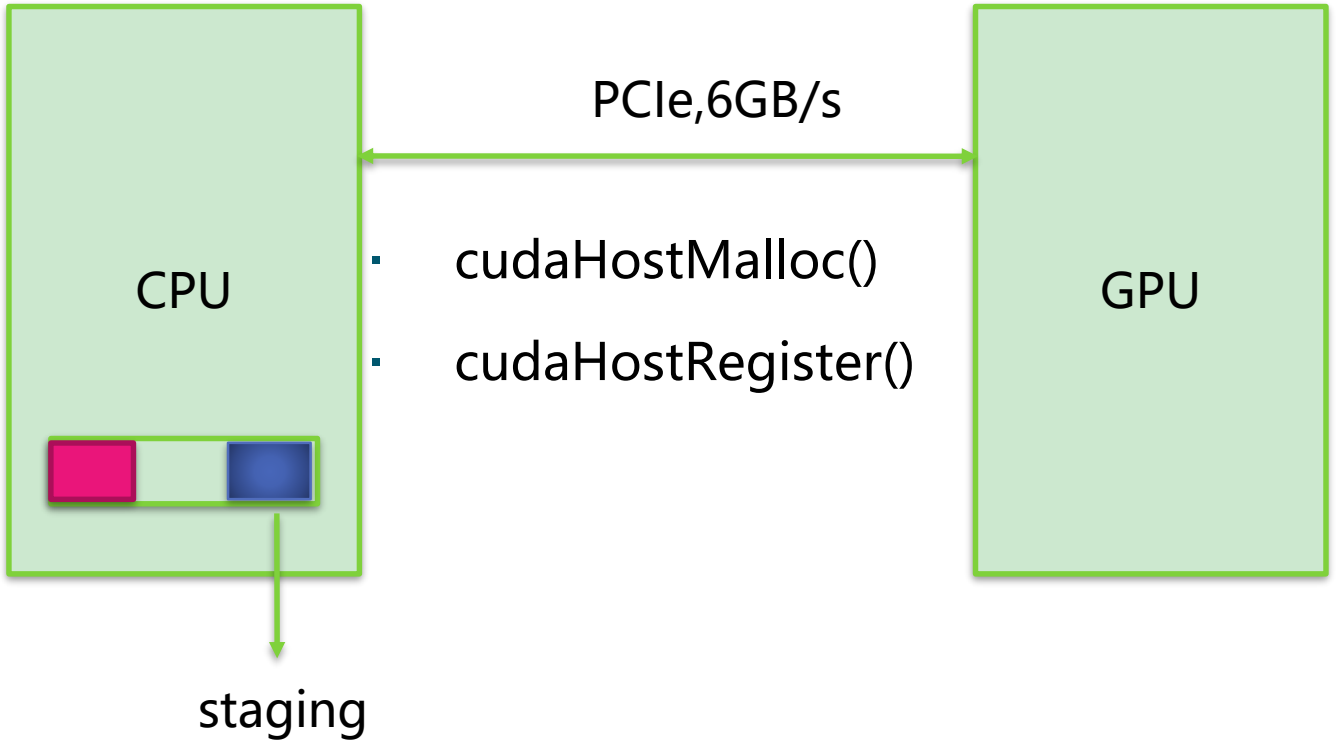
  1）如果peer-access允许字节在最短的PCIe路径上传输

  2）如果peer-access不允许CUDA驱动通过CPU memory传输

3. 纹理操作

- 纹理内存与纹理操作介绍：见“**纹理内存与纹理操作.pdf**”
- 纹理内存的优势：
- 1.它们是被缓存的,如果它们在texture fetch 中将提供更高的带宽
- 2.它们不会像全局或常驻内存读取时受内存访问模式的约束
- 3.寻址计算时的延迟更低,从而提高随机访问数据时的性能
- 4. 在一个操作中,包装的数据可以通过广播到不同的变量中
- 5.8-bit和16-bit的整型输入数据可以被转换成在范围[0.0,1.0]或[-1.0,1.0]的浮点数

4. CPU/GPU协同


PCIe,6GB/s
CPU
GPU
cudaHostMalloc()
cudaHostRegister()
staging

本周作业

- 1. 请把上周的作业并行化方法实现图像的水平翻转，用至少两个流Streams的形式优化实现。

法律声明

【声明】本视频和幻灯片为炼数成金网络课程的教学资料，所有资料只能在课程内使用，不得在课程以外范围散播，违者将可能被追究法律和经济责任。

课程详情访问炼数成金培训网站

http://edu.dataguru.cn

炼数成金逆向收费式网络课程

- Dataguru（炼数成金）是专业数据分析网站，提供教育，媒体，内容，社区，出版，数据分析业务等服务。我们的课程采用新兴的互联网教育形式，独创地发展了逆向收费式网络培训课程模式。既继承传统教育重学习氛围，重竞争压力的特点，同时又发挥互联网的威力打破时空限制，把天南地北志同道合的朋友组织在一起交流学习，使到原先孤立的学习个体组合成有组织的探索力量。并且把原先动辄成千上万的学习成本，直线下降至百元范围，造福大众。我们的目标是：低成本传播高价值知识，构架中国第一的网上知识流转阵地。
- 关于逆向收费式网络的详情，请看我们的培训网站 http://edu.dataguru.cn

Thanks

FAQ时间