CUDA高性能科学计算

2023-2024-1 秋季

公选-06110

Lecture 5

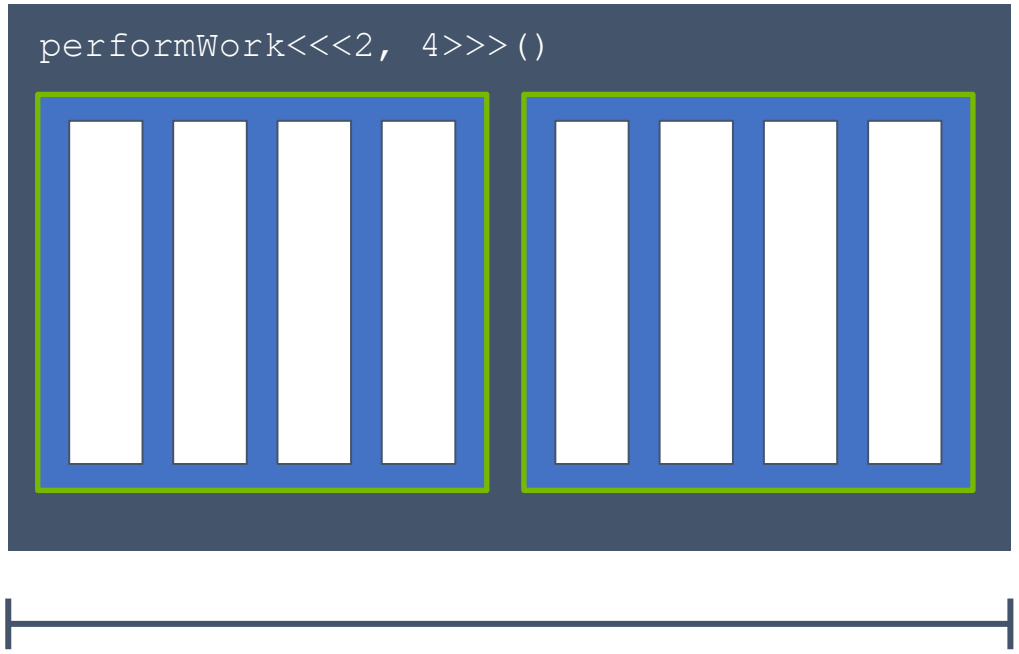
理学院　赖欣

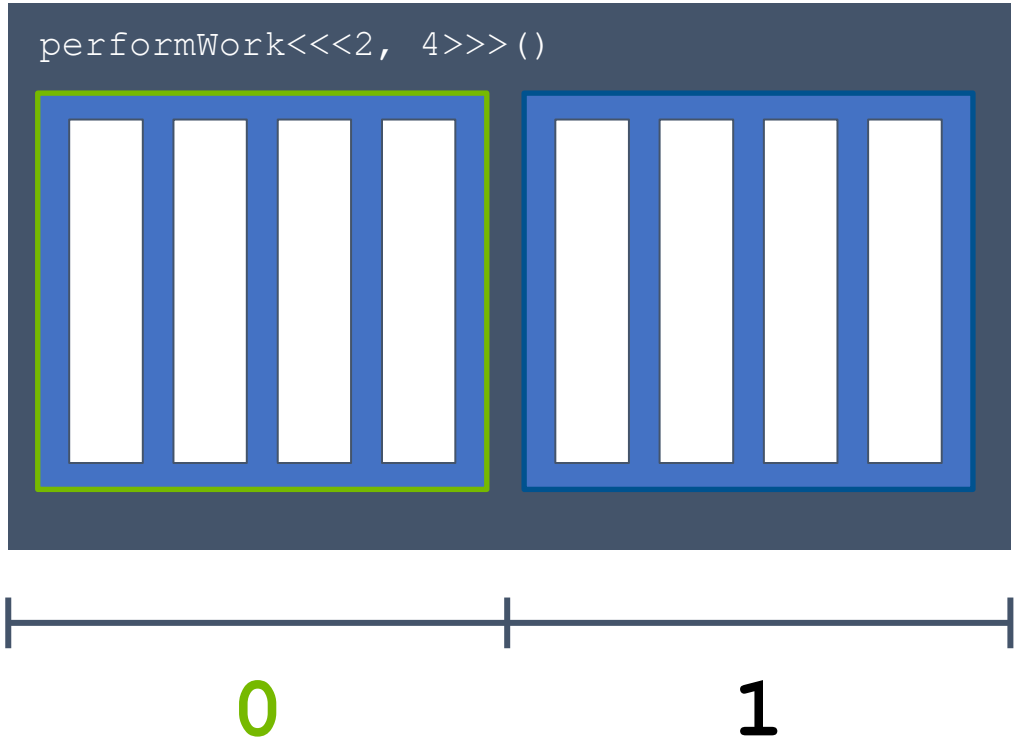
授课平台/讨论QQ群

CUDA高性能科学计算(GX)
课程编号:107016

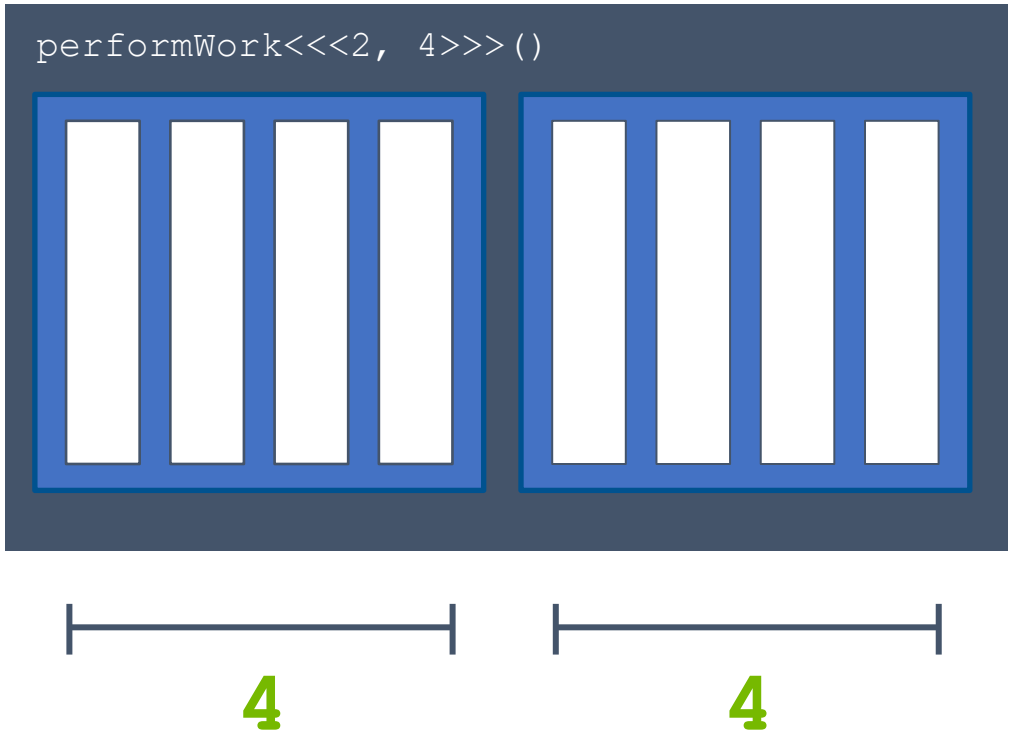

加速计算基础——CUDA C/C++
8 学时 | 中文 | 90 美元 | C/C++，CUDA ®
有培训证书

CUDA 提供的线程层次结构变量

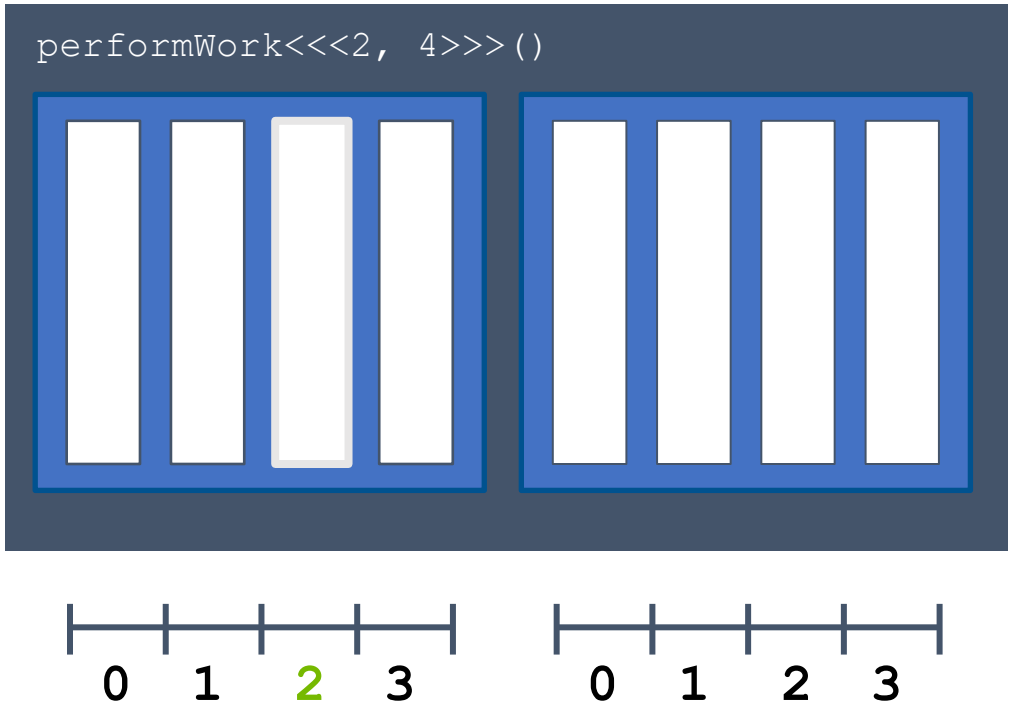




CUDA 提供的线程层次结构变量



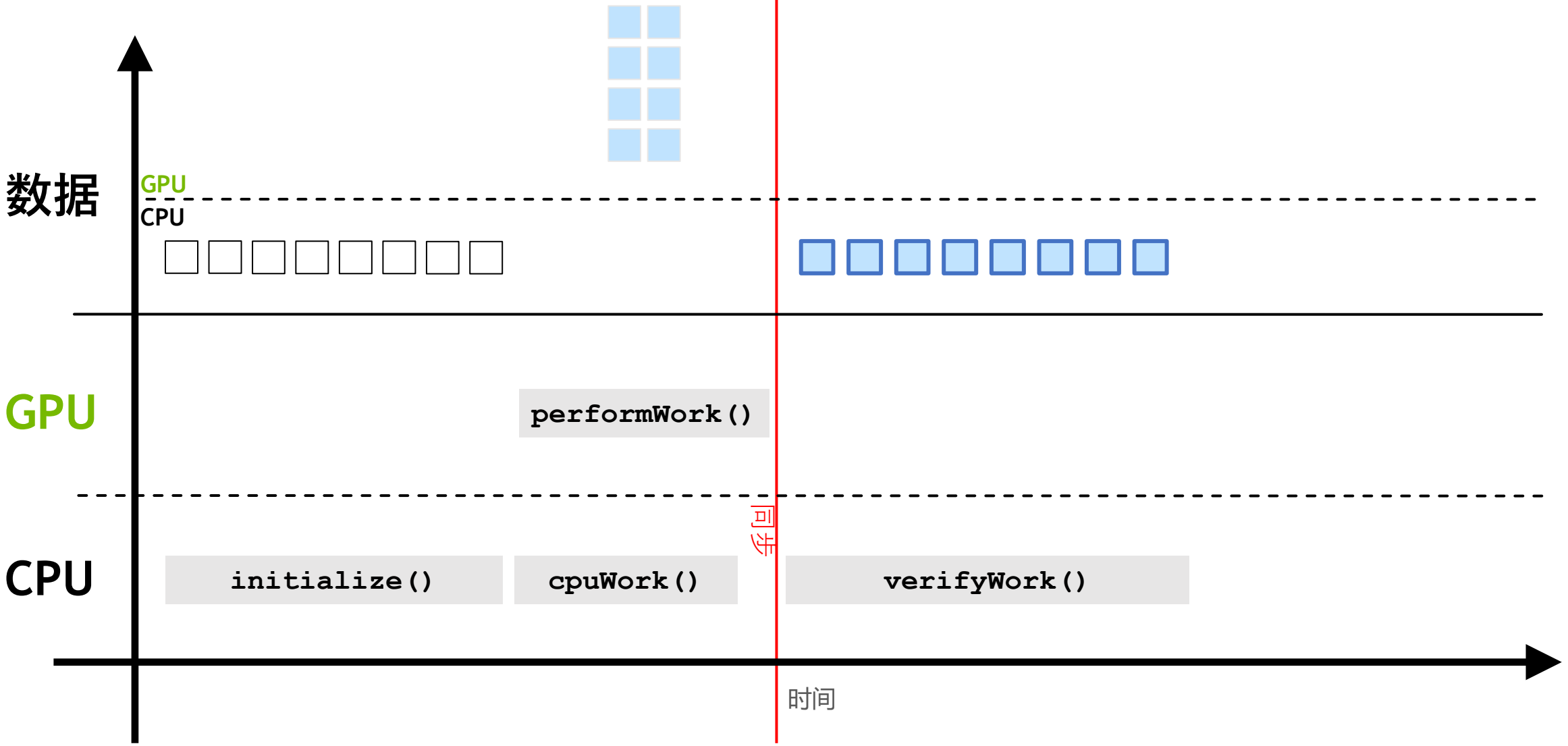
blockDim.x



threadIdx.x

blockIdx.x*blockDim.x + threadIdx.x

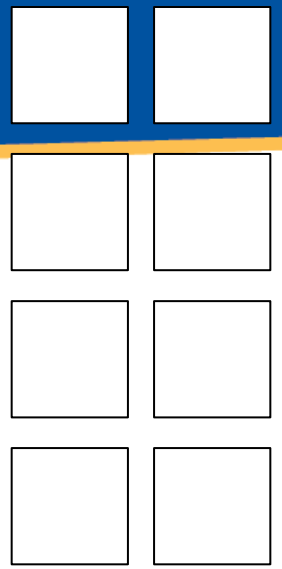
协调并行线程


数据
GPU
CPU
GPU
performWork()
同步
CPU
initialize()
cpuWork()
verifyWork()
时间

协调并行线程

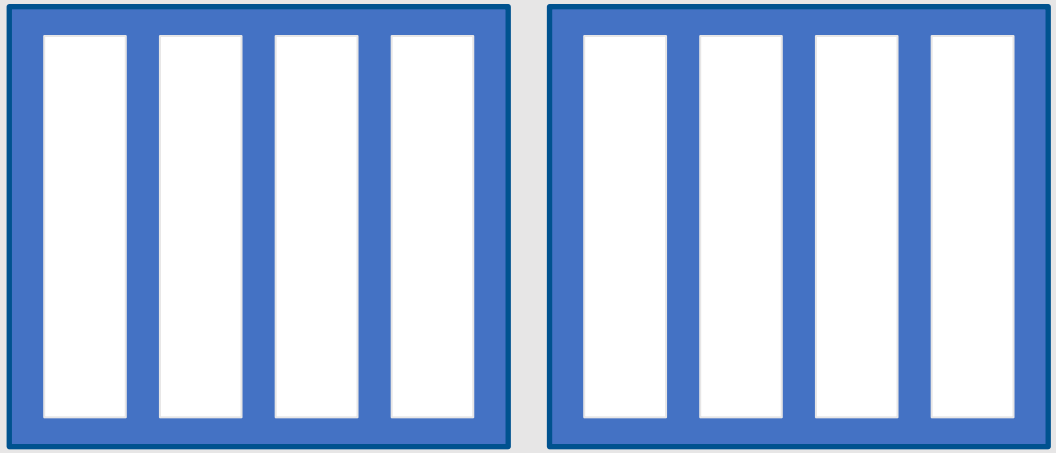
假设数据位于索引为 0 的向量中

GPU
数据



GPU

```
performWork<<<2, 4>>>()
```



协调并行线程

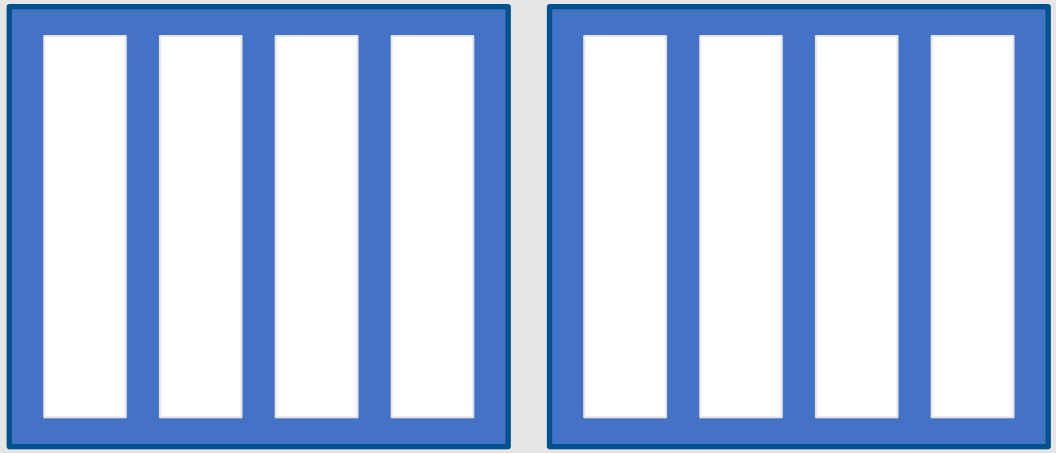
假设数据位于索引为 0 的向量中

GPU
数据

0	4
1	5
2	6
3	7

GPU

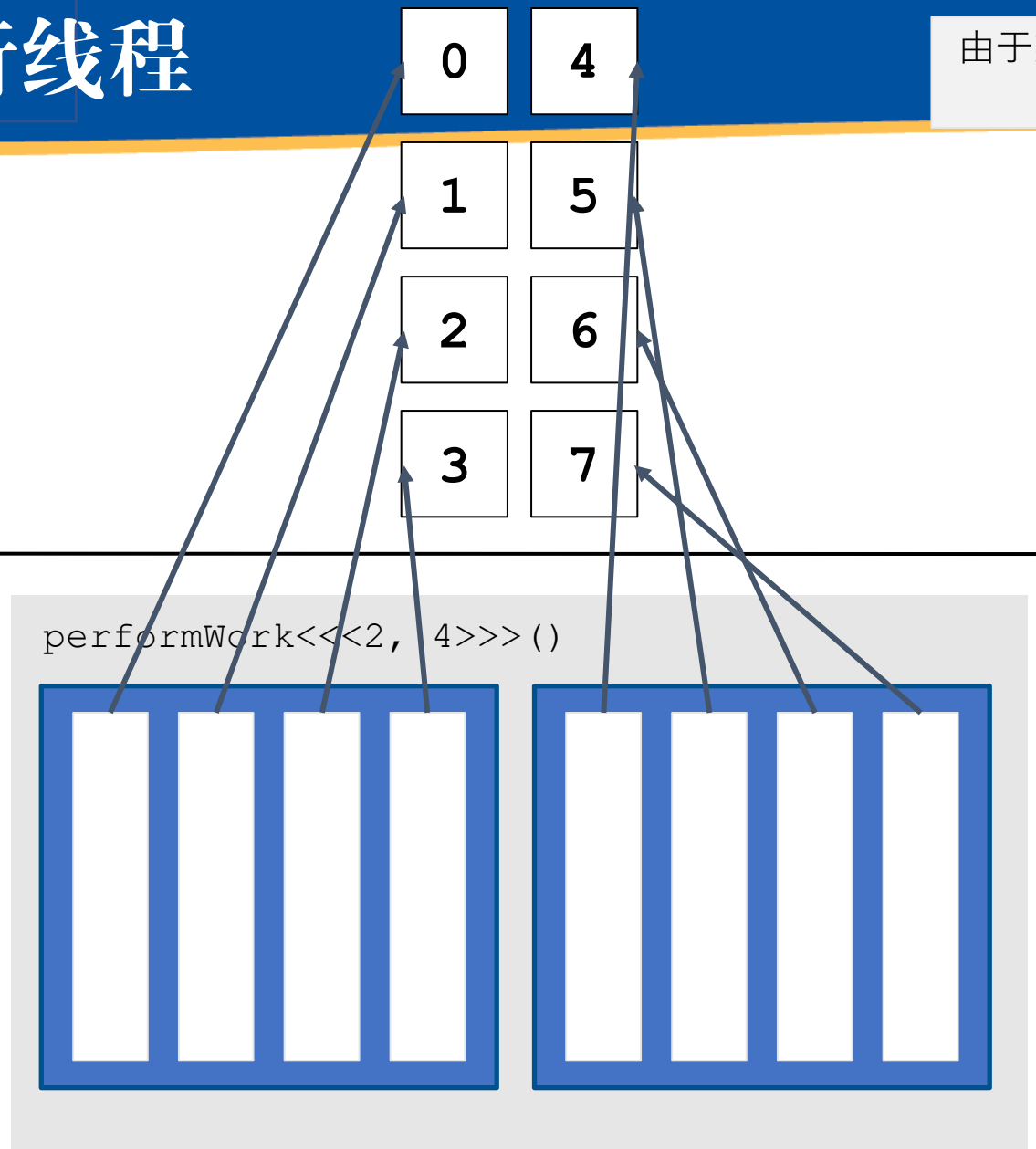
```
performWork<<<2, 4>>>()
```



协调并行线程

由于某种未知原因，必须映射每个线程以处理向量中的元素

GPU
数据



GPU

协调并行线程

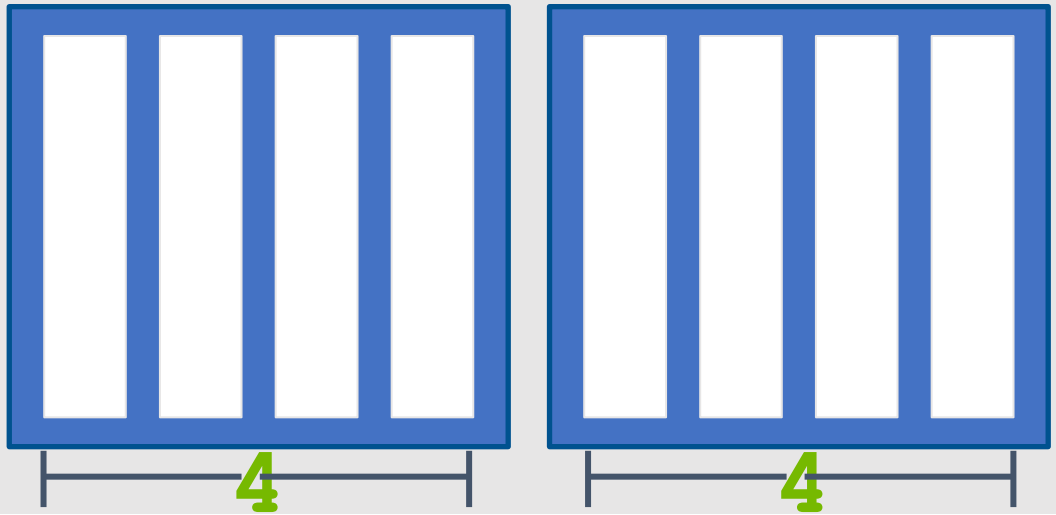
回想一下，每个线程都可以通过 **blockDim.x** 访问所在块的大小

GPU
数据

0	4
1	5
2	6
3	7

GPU

```
performWork<<<2, 4>>>()
```



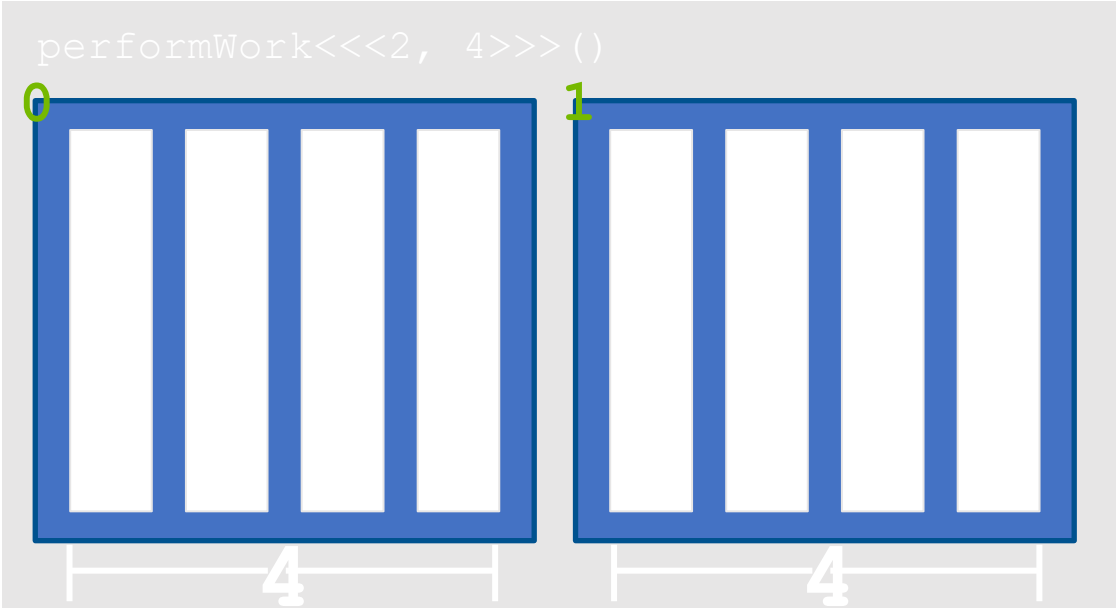
协调并行线程

0	4
1	5
2	6
3	7

…并通过 **blockIdx.x** 访问网格内其所在块的索引

GPU
数据

GPU



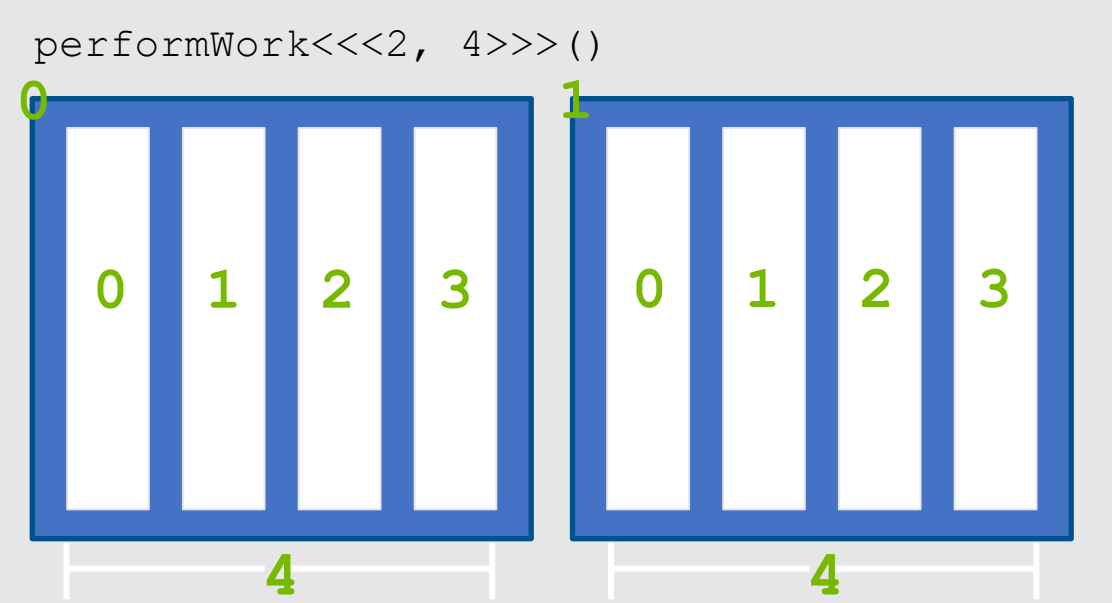
协调并行线程

0	4
1	5
2	6
3	7

…并通过 **threadIdx.x** 访问所在块内自身的线程索引

GPU
数据

GPU



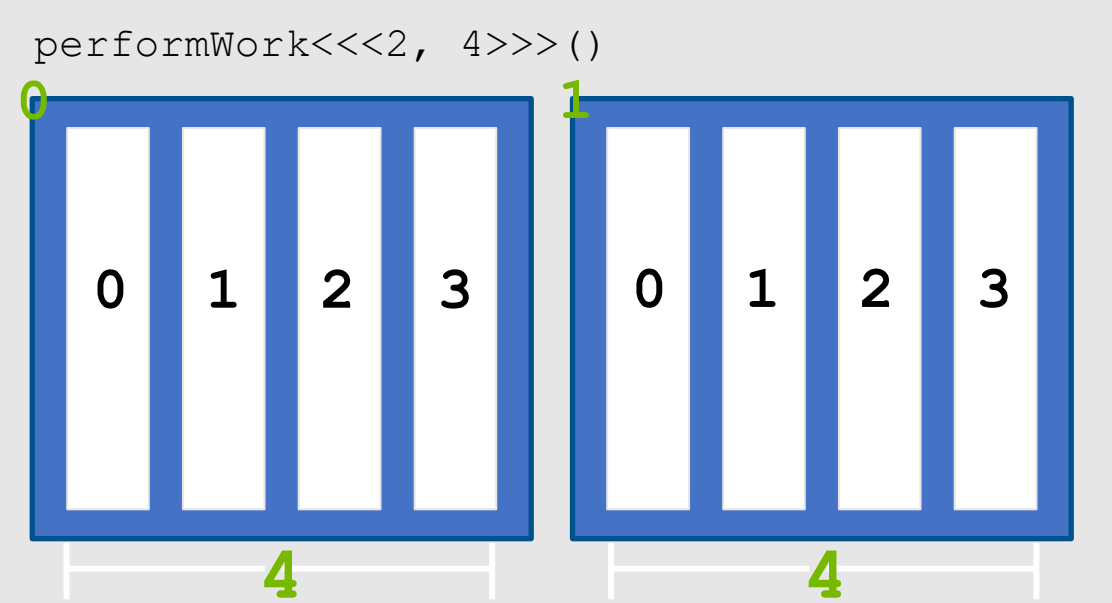
协调并行线程

GPU
数据

0	4
1	5
2	6
3	7

通过这些变量，公式 **threadIdx.x + blockIdx.x * blockDim.x** 可将每个线程映射到向量的元素中

GPU



协调并行线程

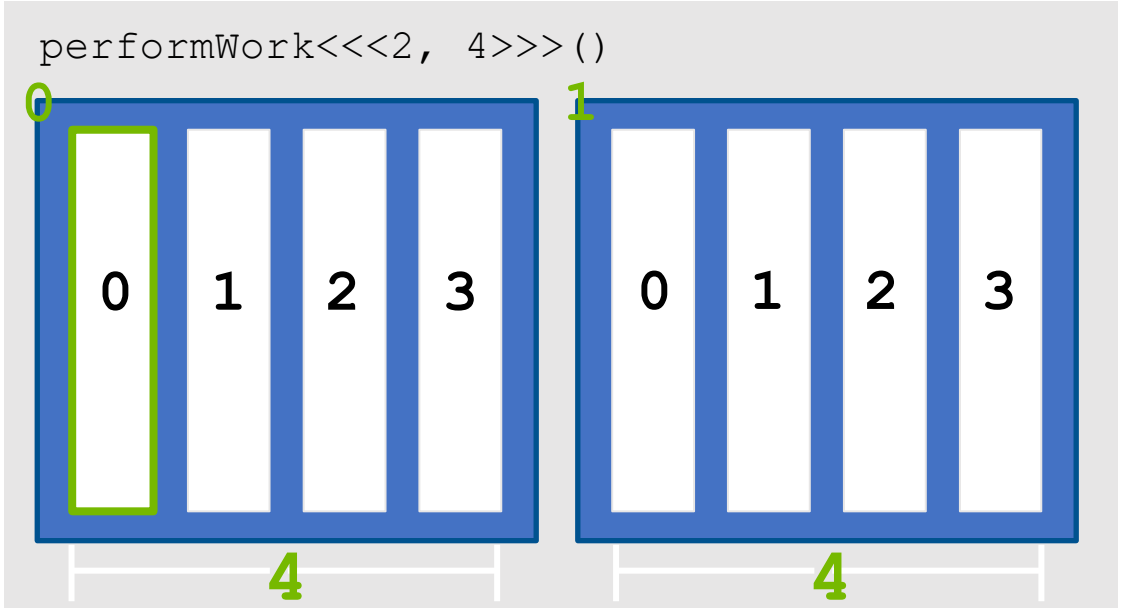
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
0		0		4

dataIndex
?

GPU



协调并行线程

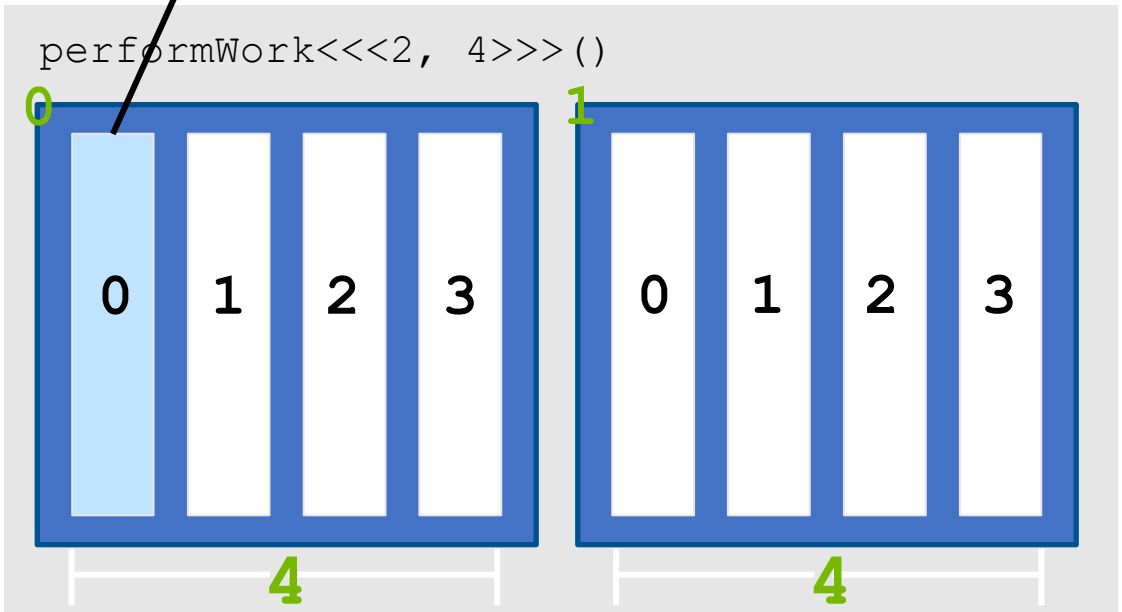
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
0		0		4

dataIndex
0

GPU



协调并行线程

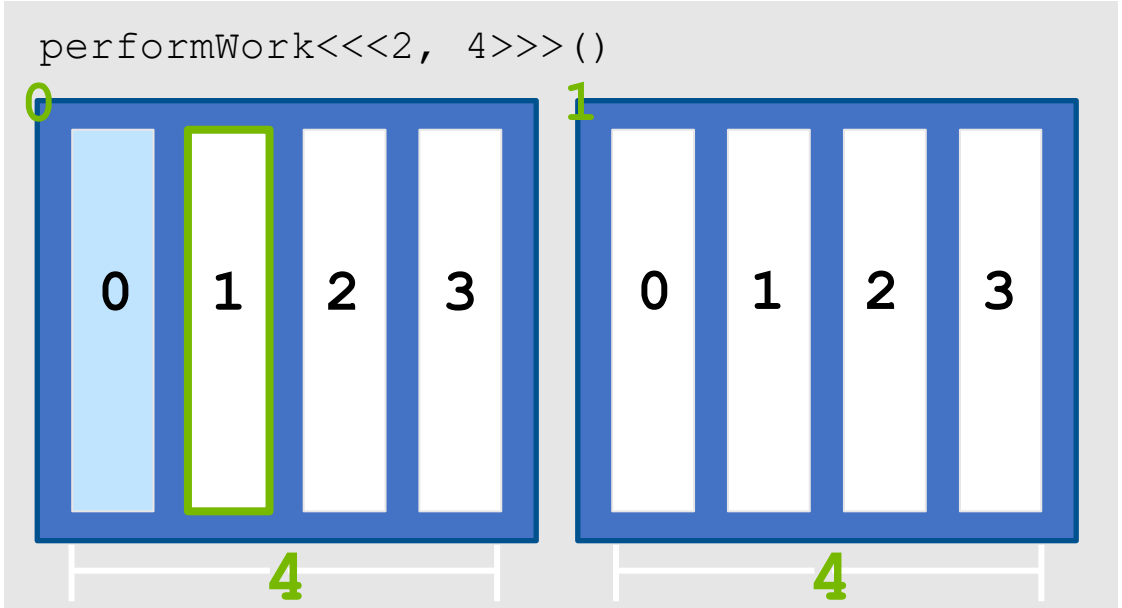
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		0		4

dataIndex
?

GPU



协调并行线程

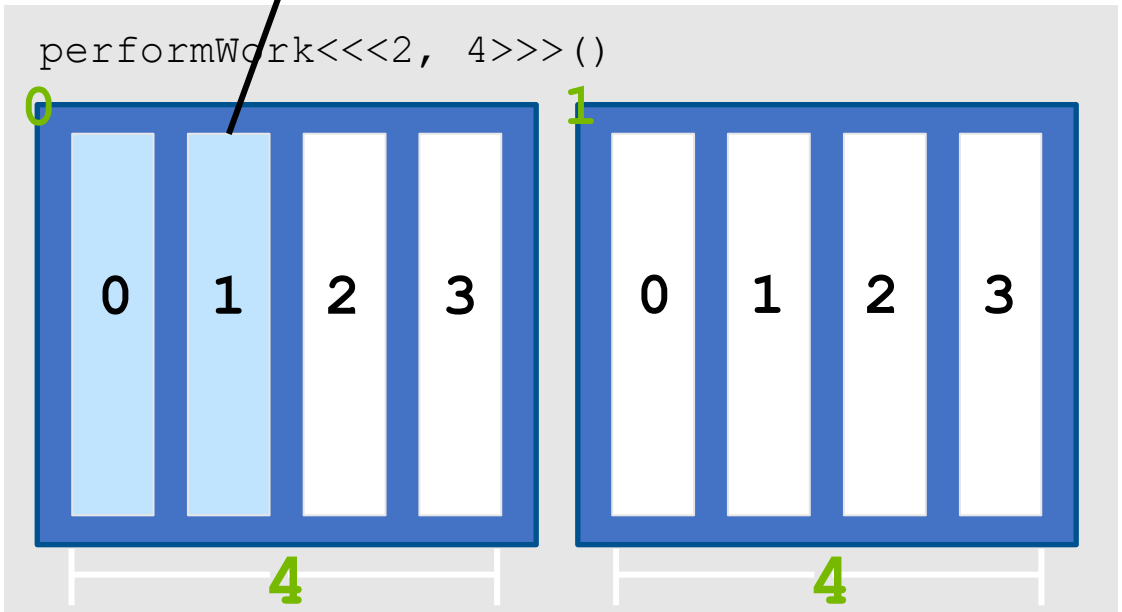
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		0		4

dataIndex
1

GPU



协调并行线程

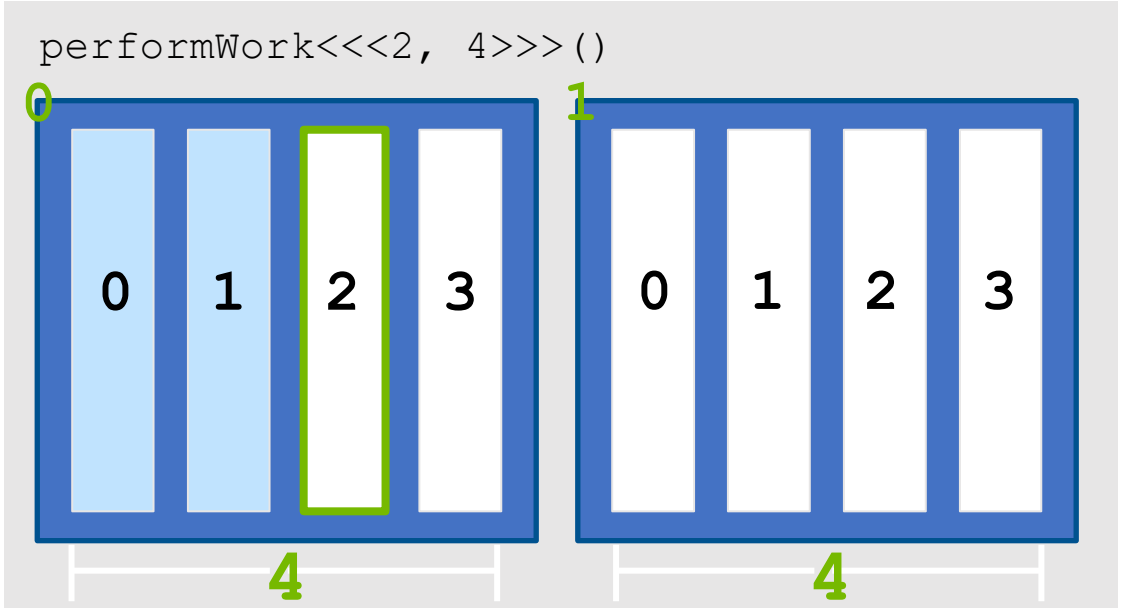
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		0		4

dataIndex
?

GPU



协调并行线程

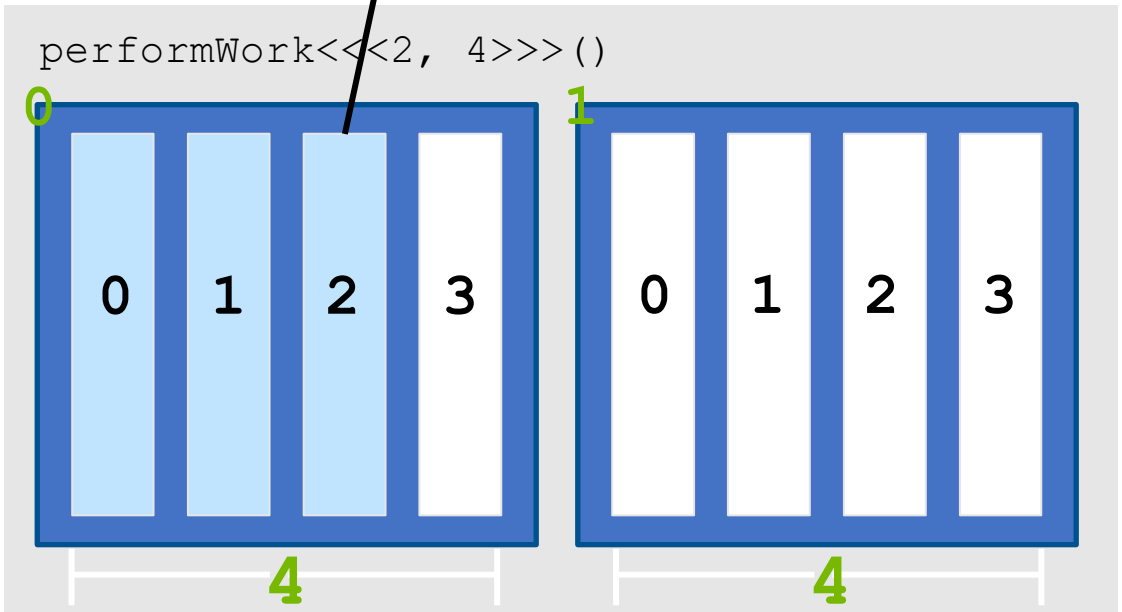
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		0		4

dataIndex
2

GPU



协调并行线程

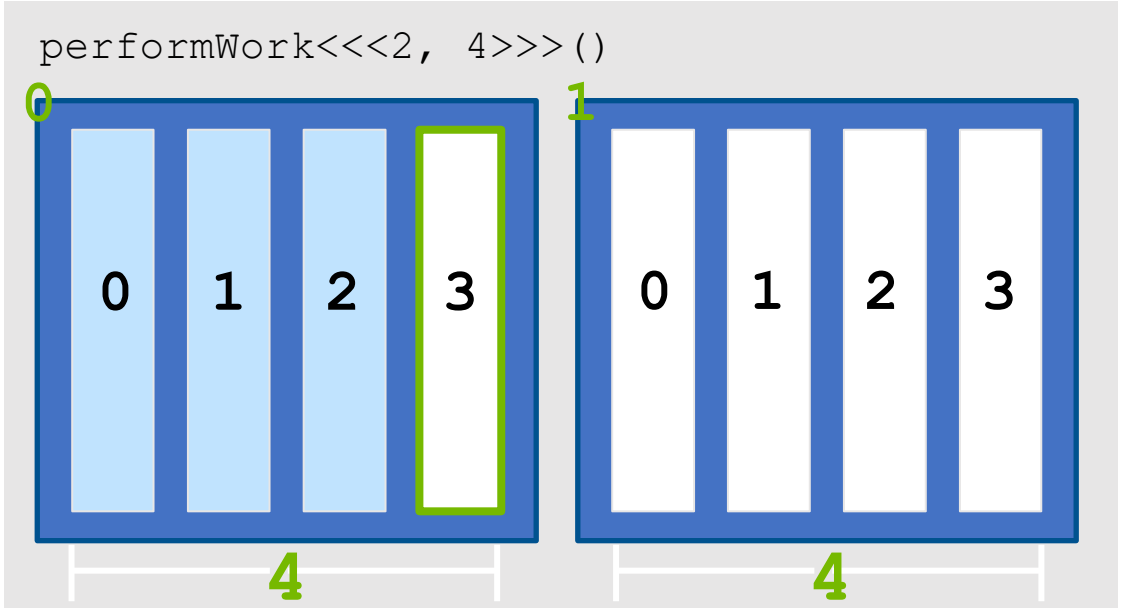
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
3		0		4

dataIndex
?

GPU



协调并行线程

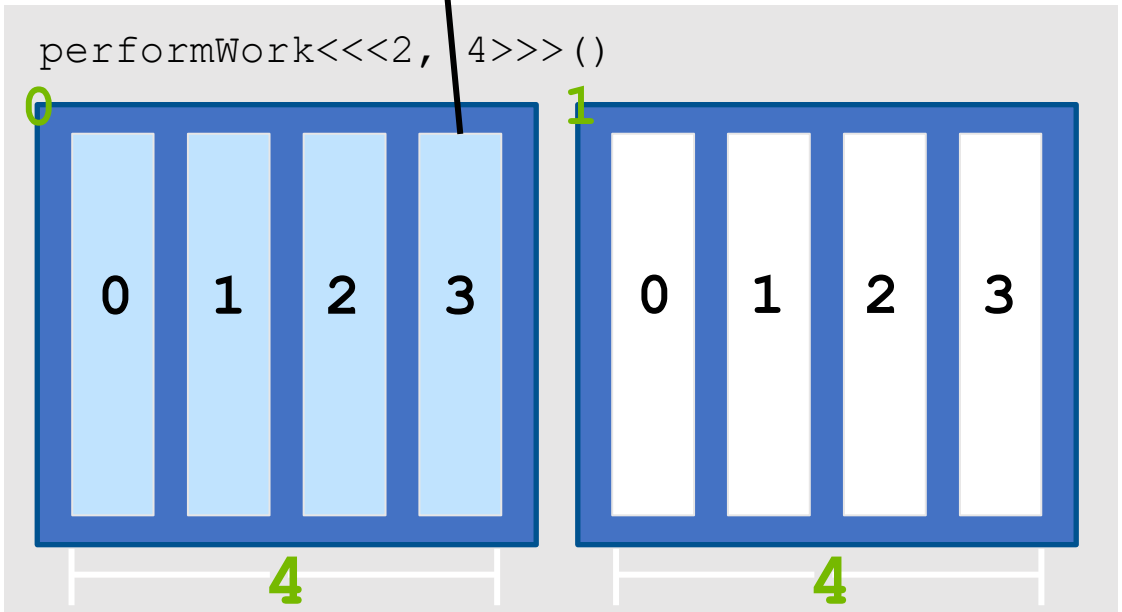
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
3		0		4

dataIndex
3

GPU



协调并行线程

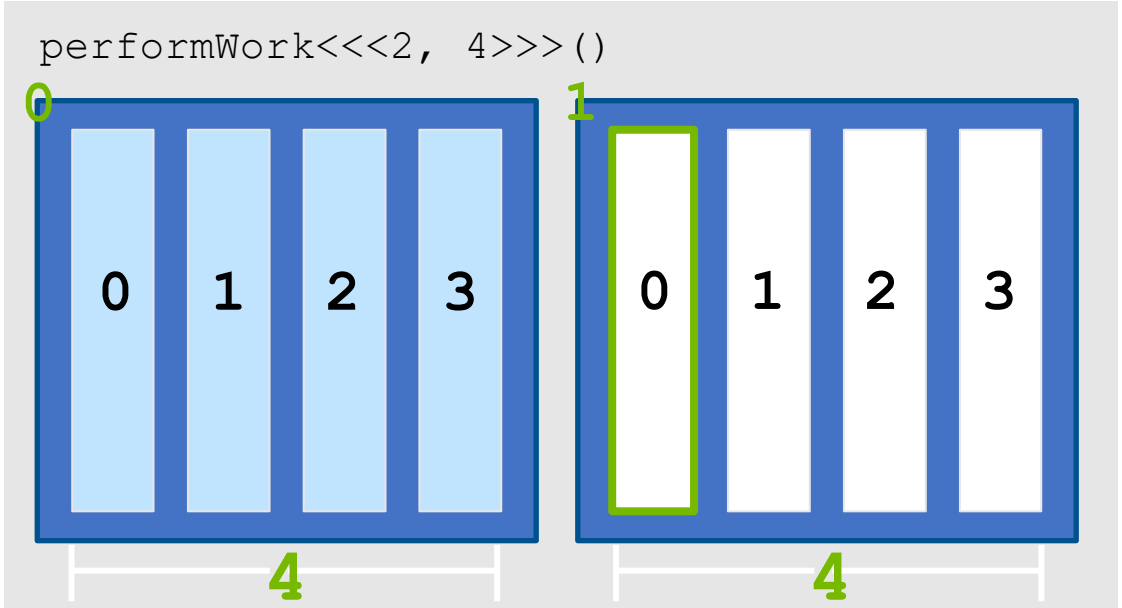
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
0		1		4

dataIndex
?

GPU



协调并行线程

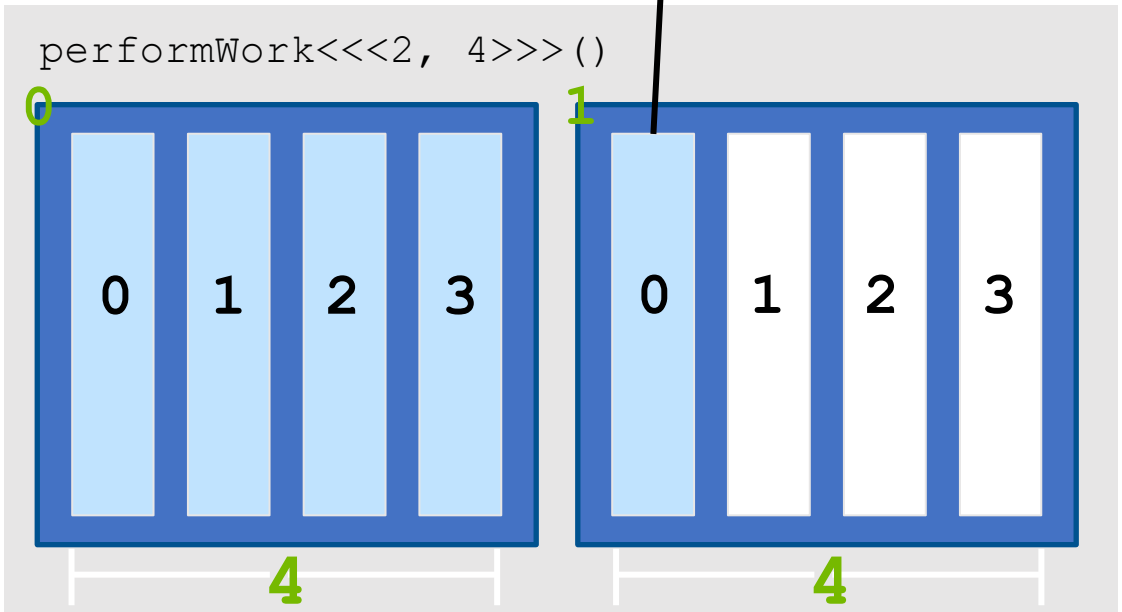
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
0		1		4

dataIndex
4

GPU



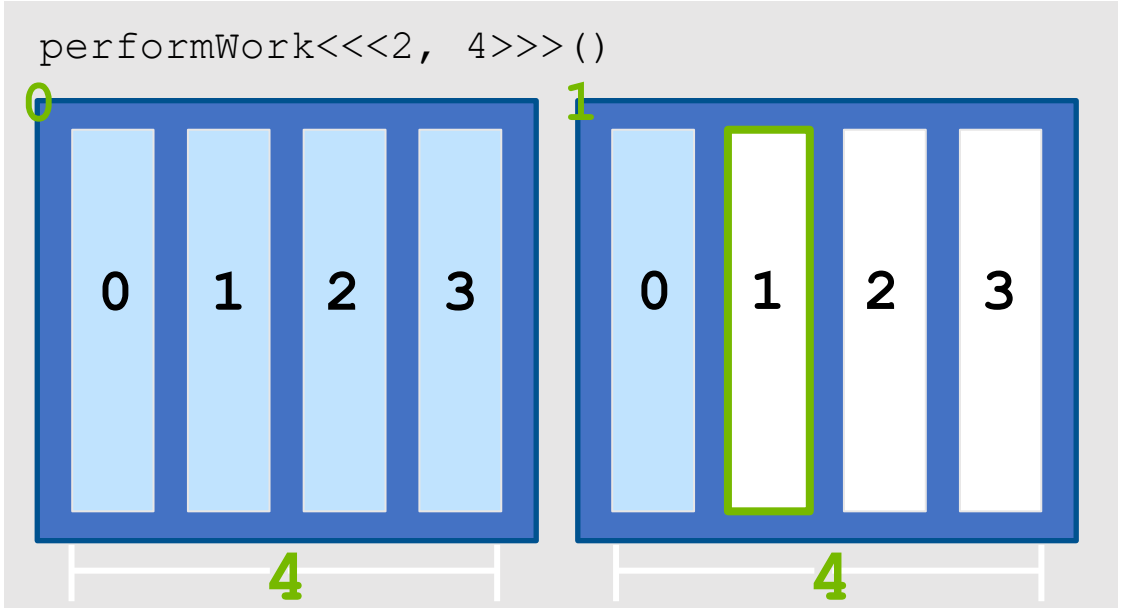
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		1		4

dataIndex
?

GPU



协调并行线程

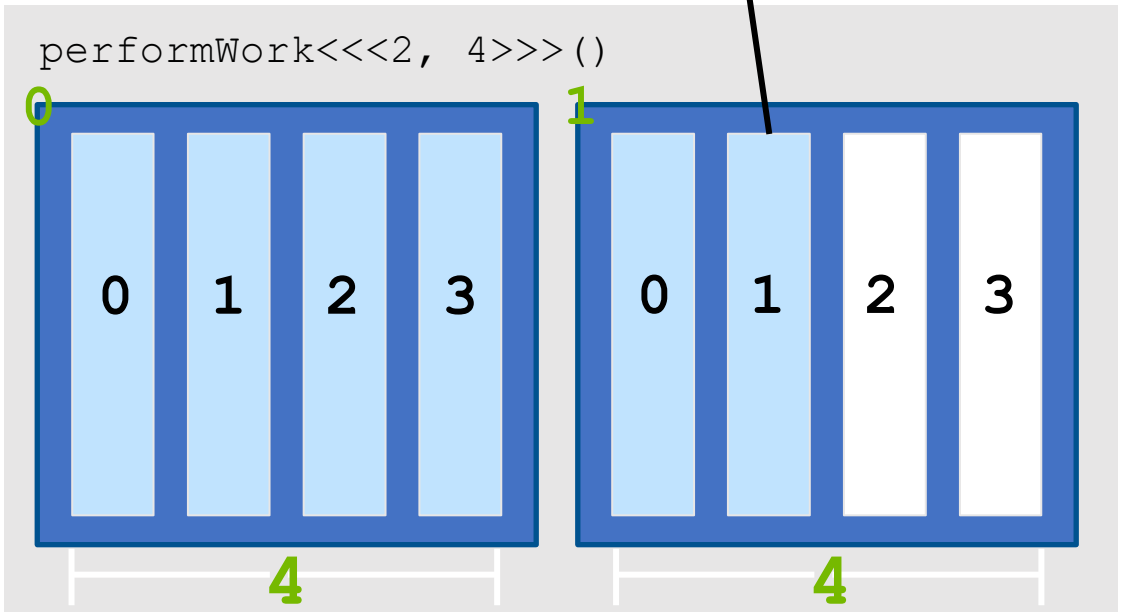
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		1		4

dataIndex
5

GPU



协调并行线程

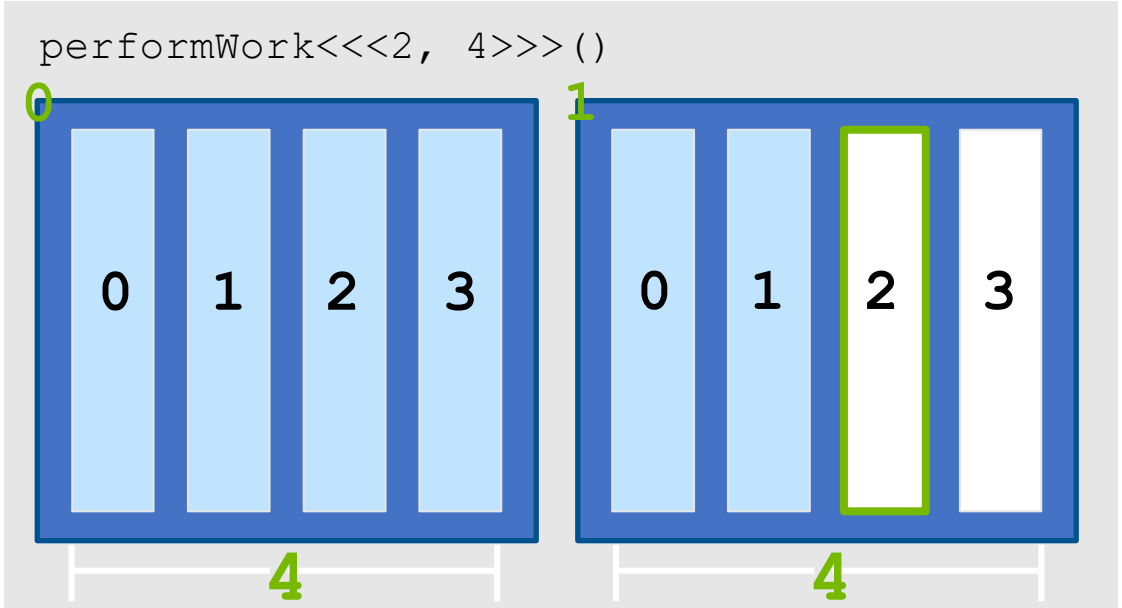
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4

dataIndex
?

GPU



协调并行线程

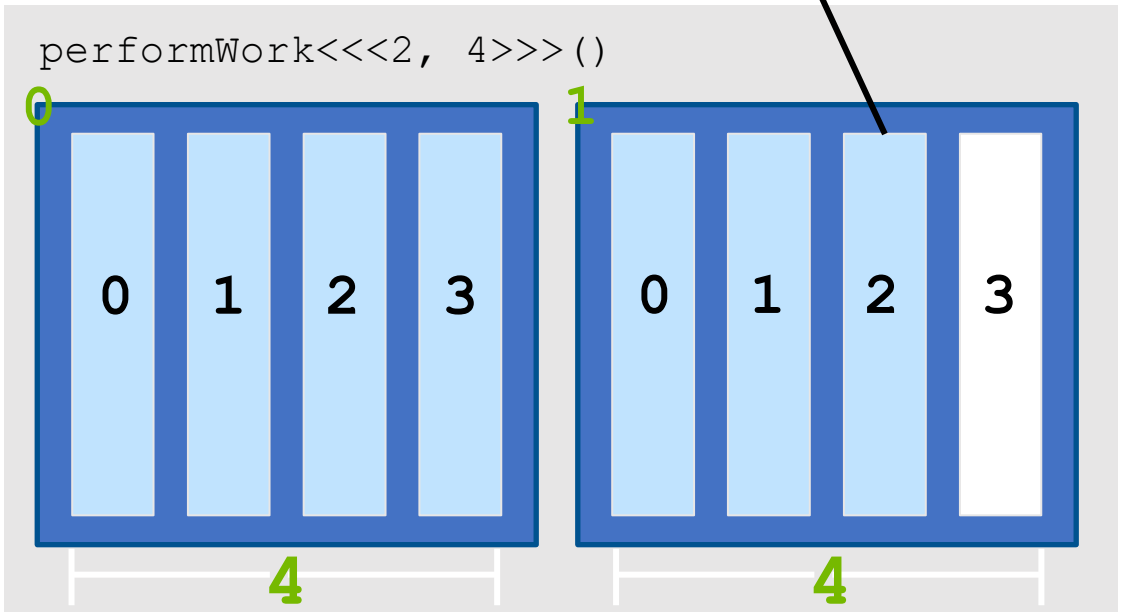
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4

dataIndex
6

GPU



协调并行线程

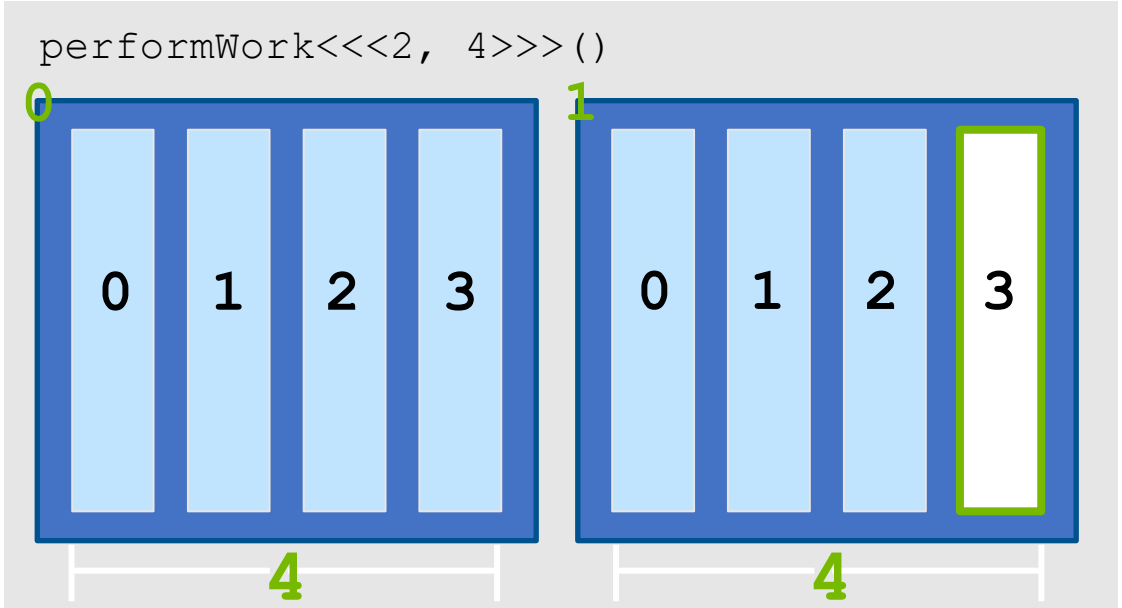
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
3		1		4

dataIndex
?

GPU



协调并行线程

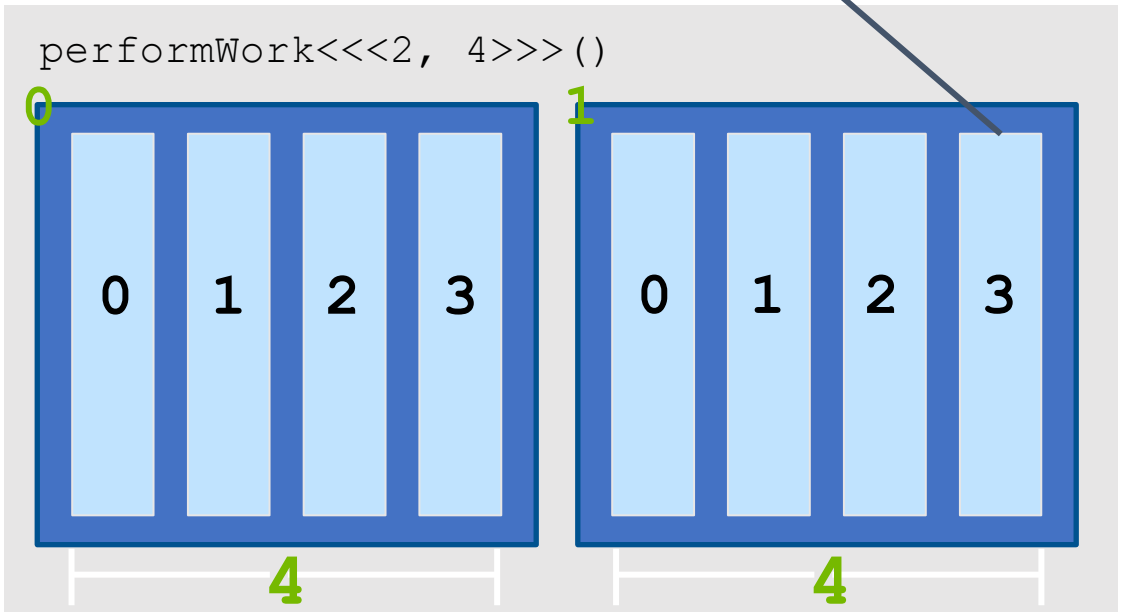
GPU
数据

0	4
1	5
2	6
3	7

threadIdx.x	+	blockIdx.x	*	blockDim.x
3		1		4

dataIndex
7

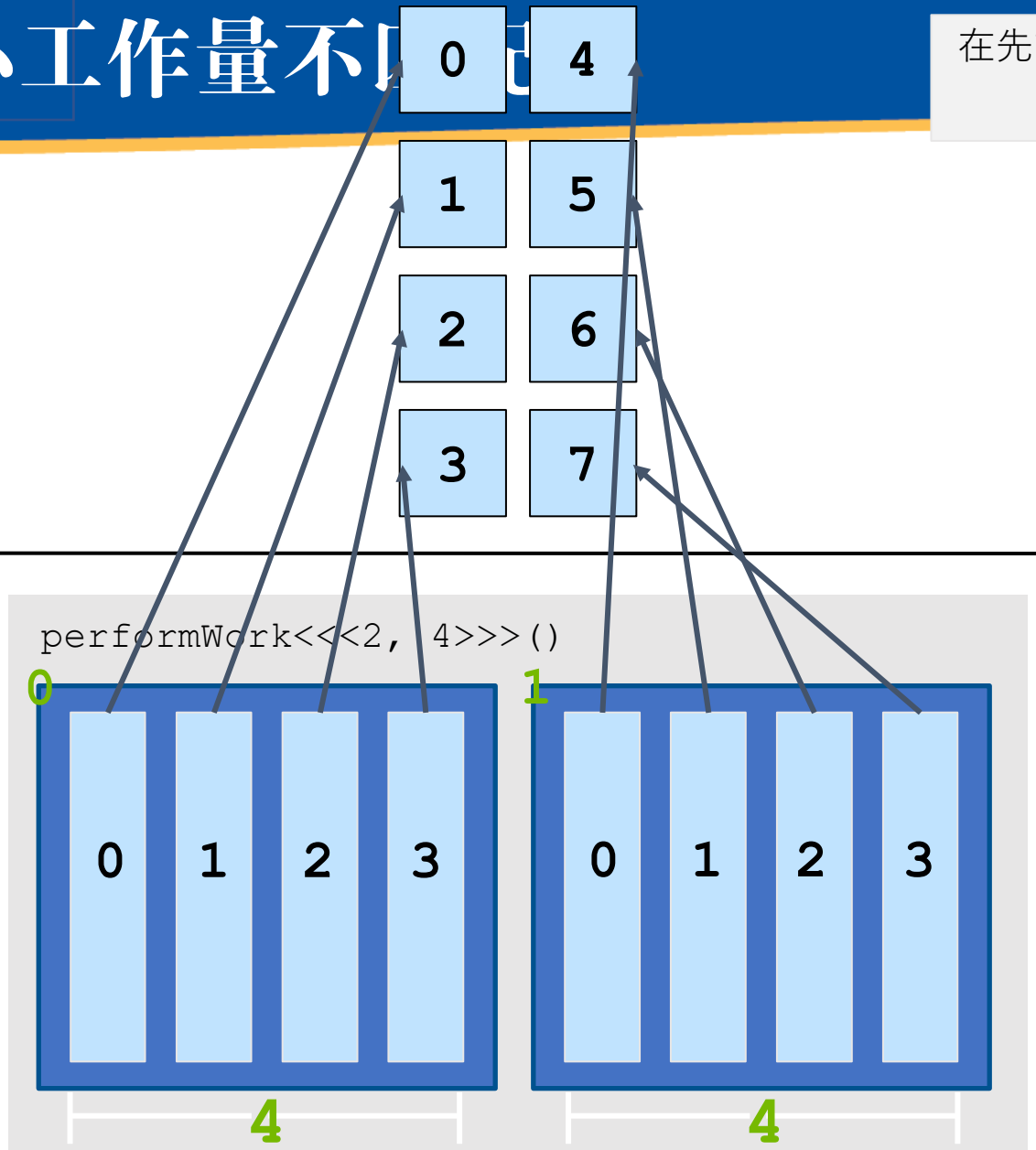
GPU



网格大小工作量不匹配

在先前场景中，网络中的线程数与元素数量完全匹配

GPU
数据

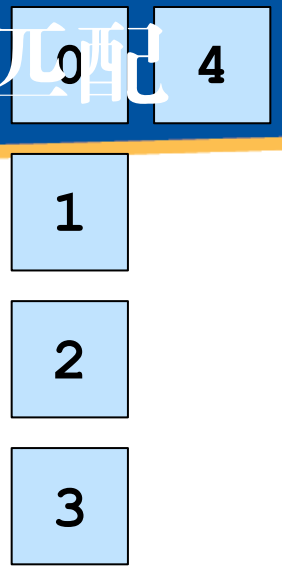


GPU

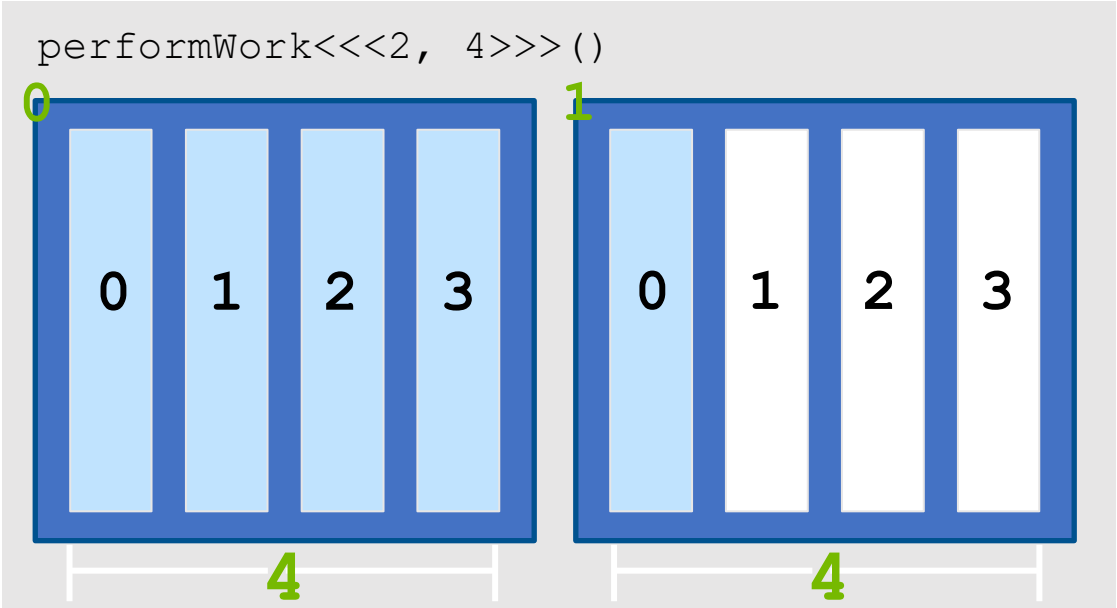
网格大小工作量不匹配

如果线程数超过要完成的工作量，
该怎么办？

GPU
数据



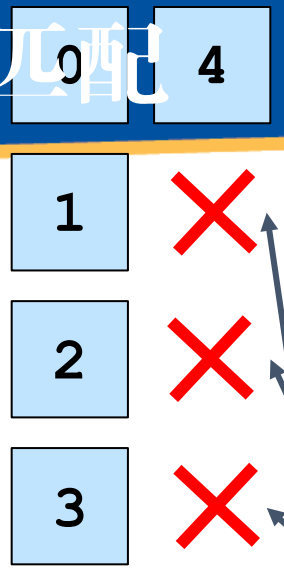
GPU



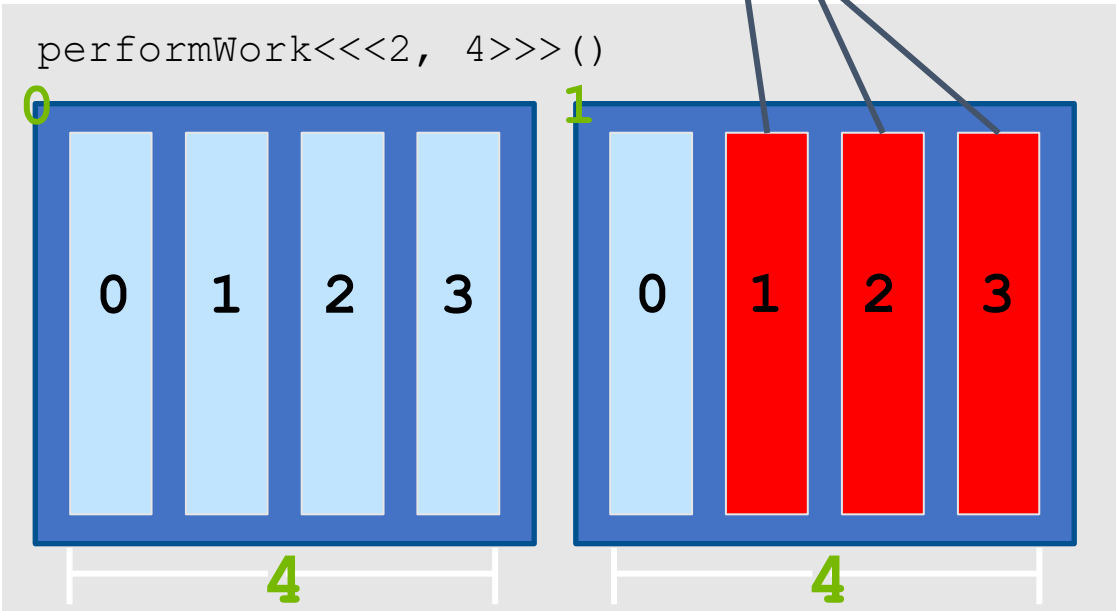
网格大小工作量不匹配

尝试访问不存在的元素会导致运行时错误

GPU
数据



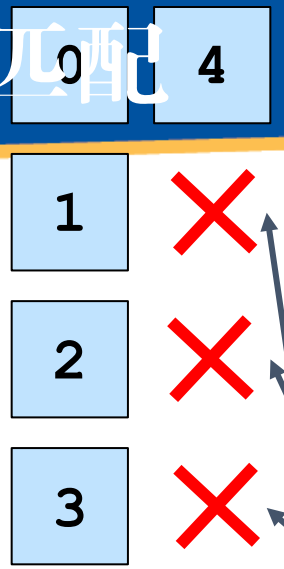
GPU



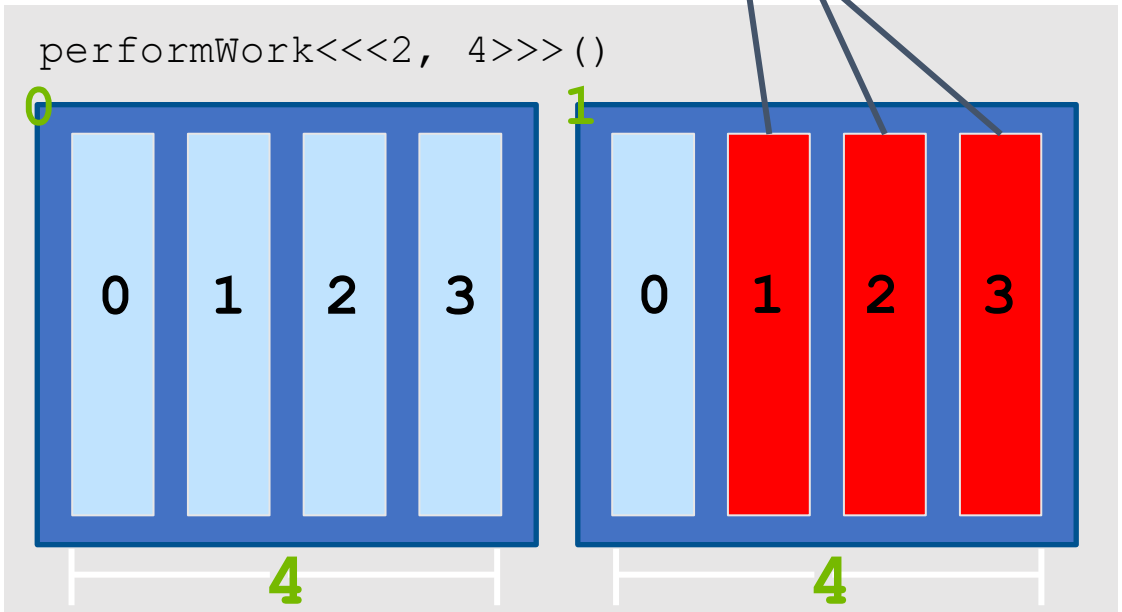
网格大小工作量不匹配

必须使用代码检查并确保经由公式 **threadIdx.x + blockIdx.x * blockDim.x** 计算出的 **dataIndex** 小于 **N**（数据元素数量）。

GPU
数据

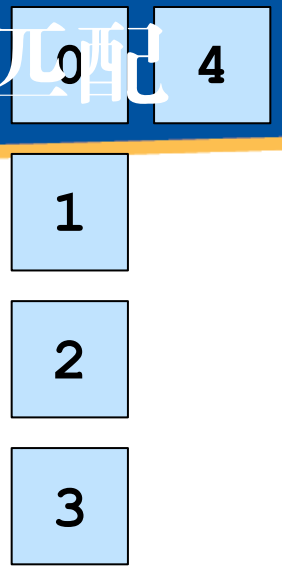


GPU



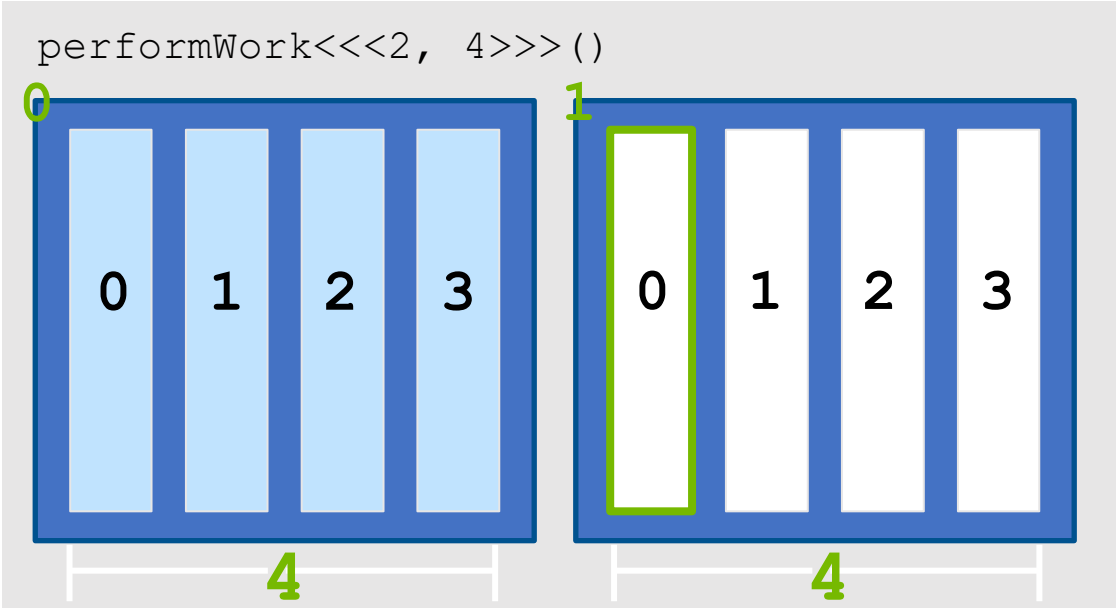
网格大小工作量不匹配

GPU
数据



threadIdx.x	+	blockIdx.x	*	blockDim.x
0		1		4
dataIndex	<	N	=	可以运行
4		5		?

GPU



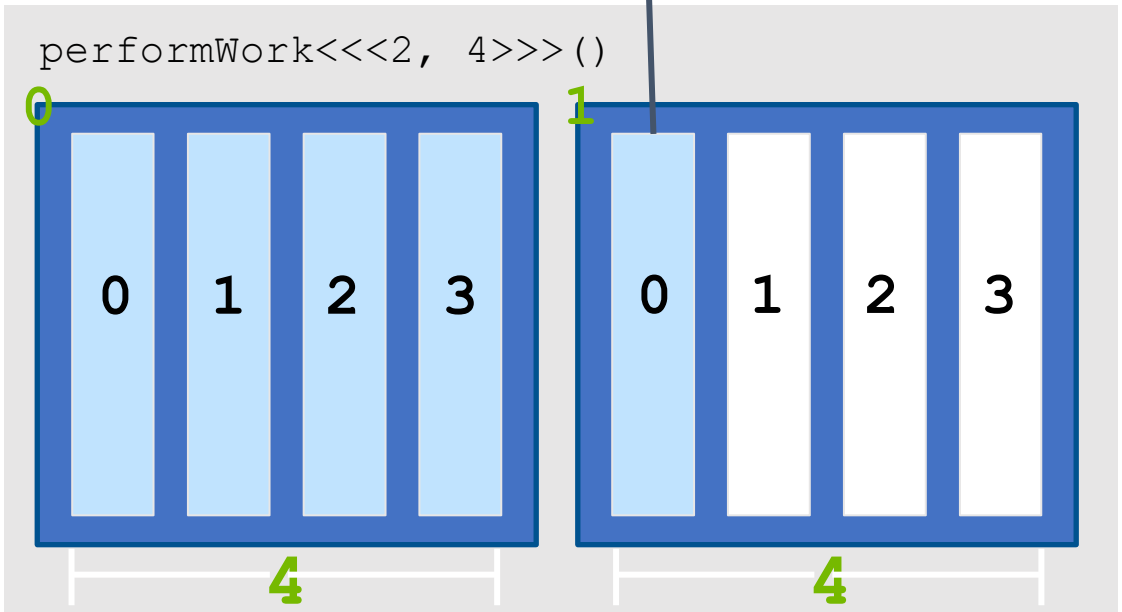
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
0		1		4
dataIndex	<	N	=	可以运行
4		5		true

GPU



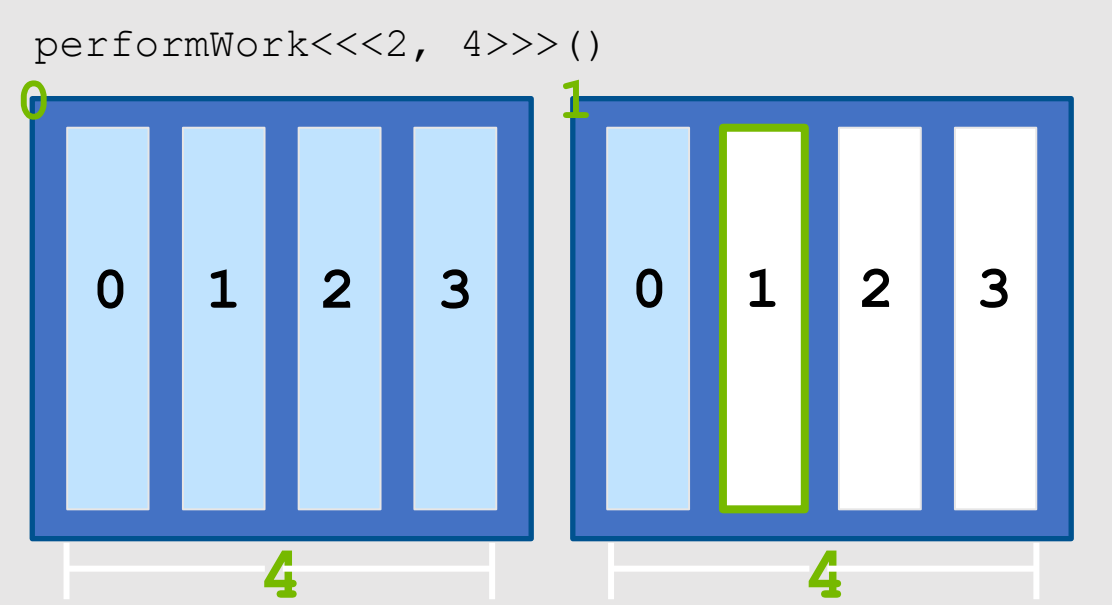
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		1		4
dataIndex	<	N	=	可以运行
5		5		?

GPU



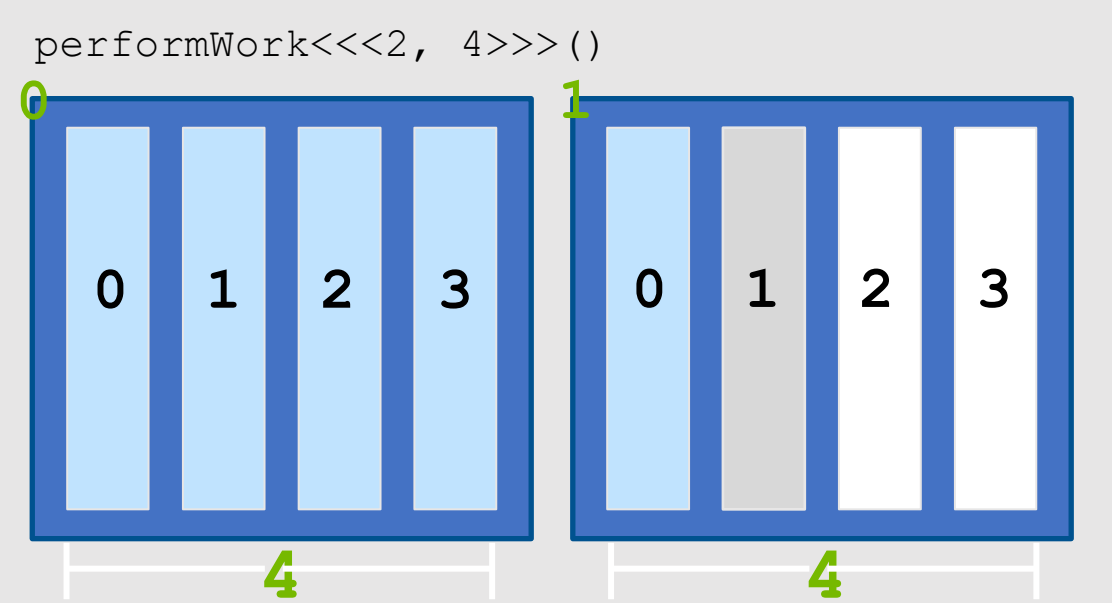
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
1		1		4
dataIndex	<	N	=	可以运行
5		5		false

GPU



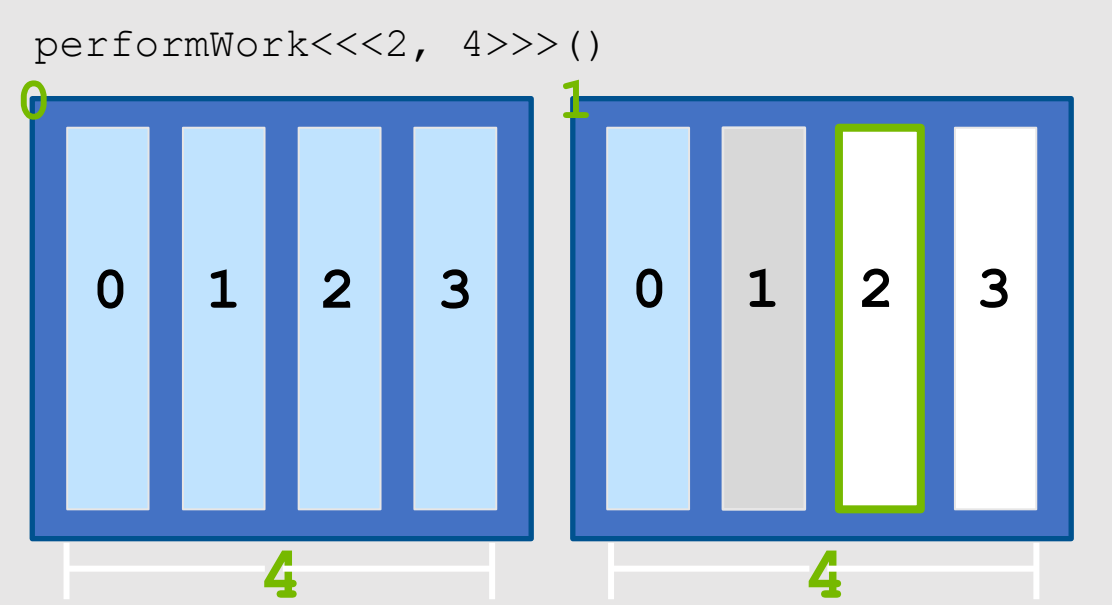
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4
dataIndex	<	N	=	可以运行
6		5		?

GPU



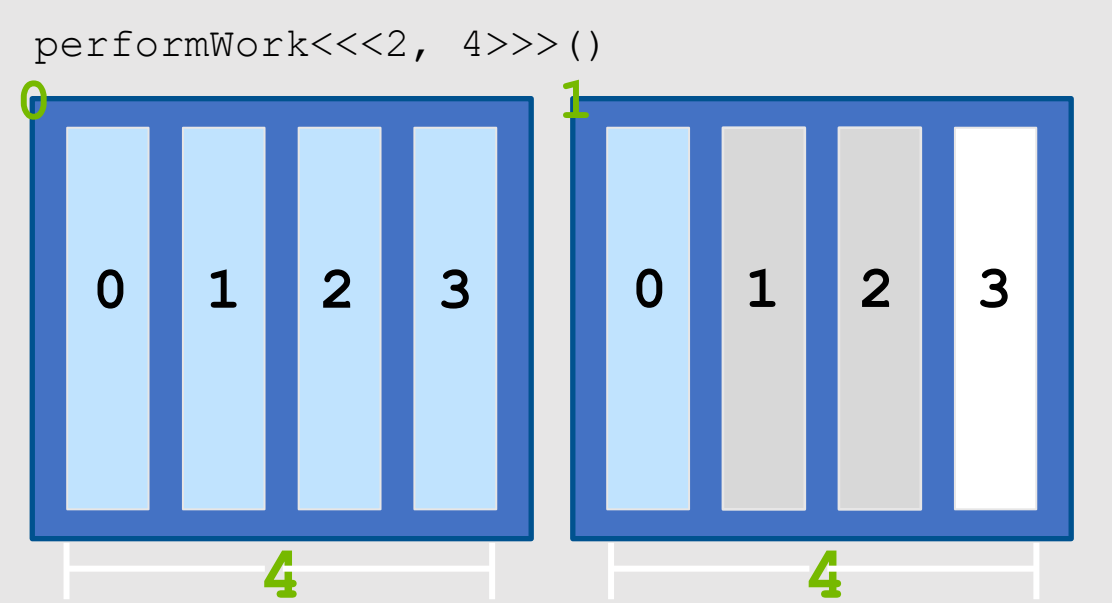
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4
dataIndex	<	N	=	可以运行
6		5		false

GPU



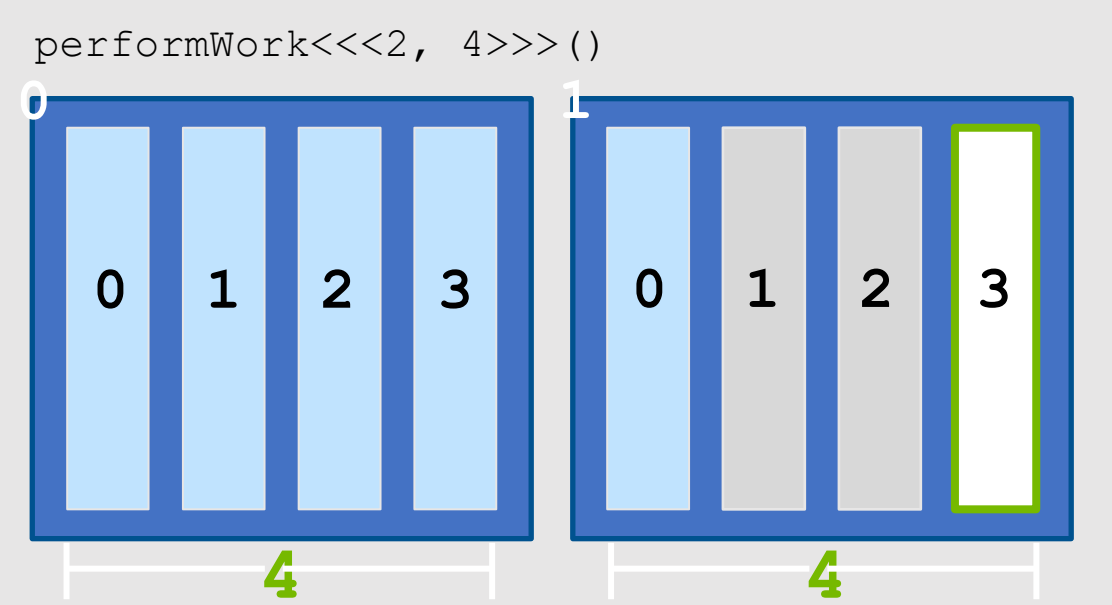
网格大小工作量不匹配

GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4
dataIndex	<	N	=	可以运行
6		5		?

GPU



网格大小工作量不匹配

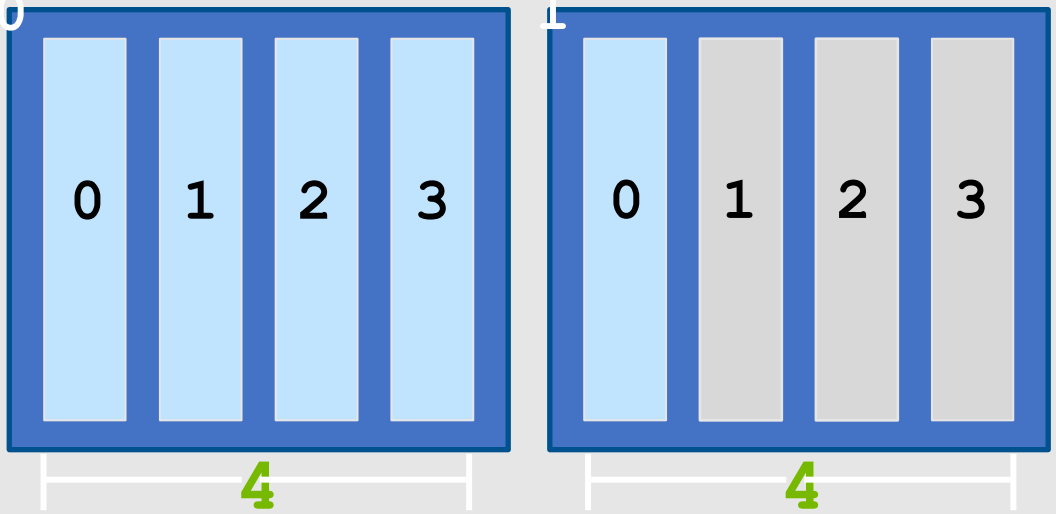
GPU
数据

0	4
1	
2	
3	

threadIdx.x	+	blockIdx.x	*	blockDim.x
2		1		4
dataIndex	<	N	=	可以运行
6		5		false

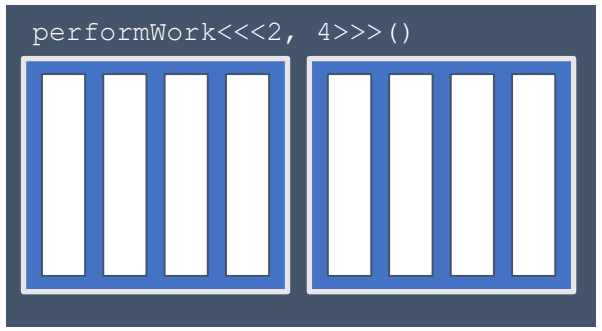
GPU

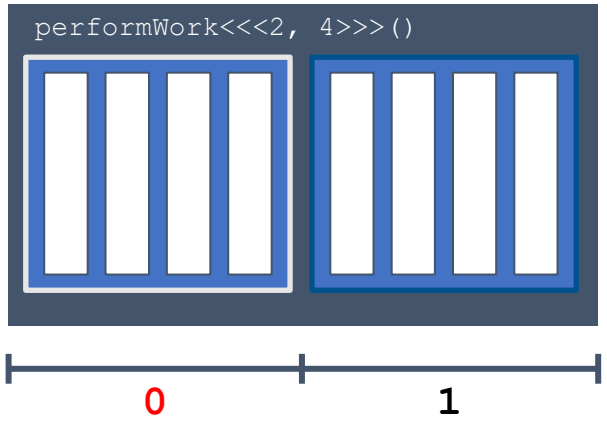
```
performWork<<<2, 4>>>()
```

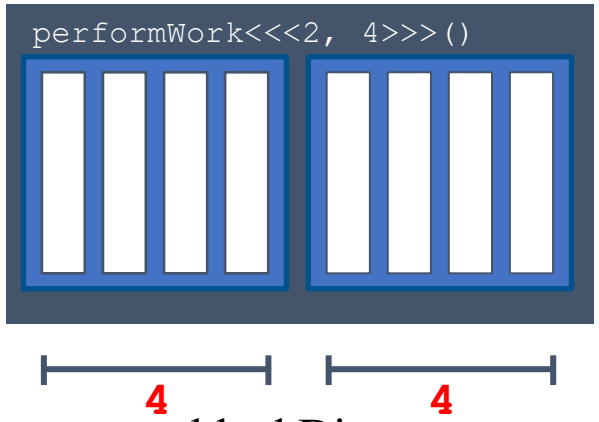


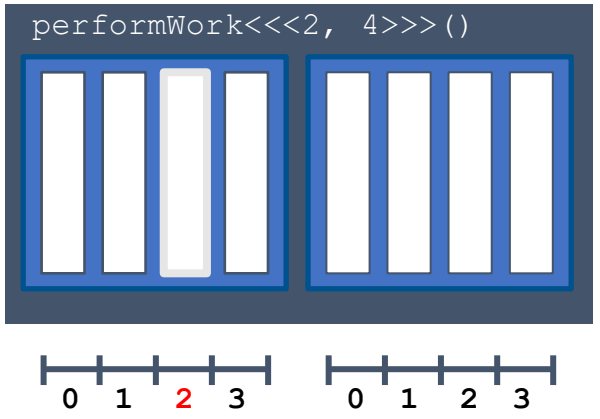
- 1000**数据**　1000/256 = 3.9　　3+1=4　　4*256=1024
- 1024**数据**　1024/256 = 4
- 1025**数据**　1025/256 = 4.01　4+1=5　　5*256=1280
- size_t number_of_blocks = (N + threads_per_block - 1) / threads_per_block;

网格与线程（一维）


performWork<<<2, 4>>>()
2
gridDim.x


performWork<<<2, 4>>>()
0
1
blockIdx.x

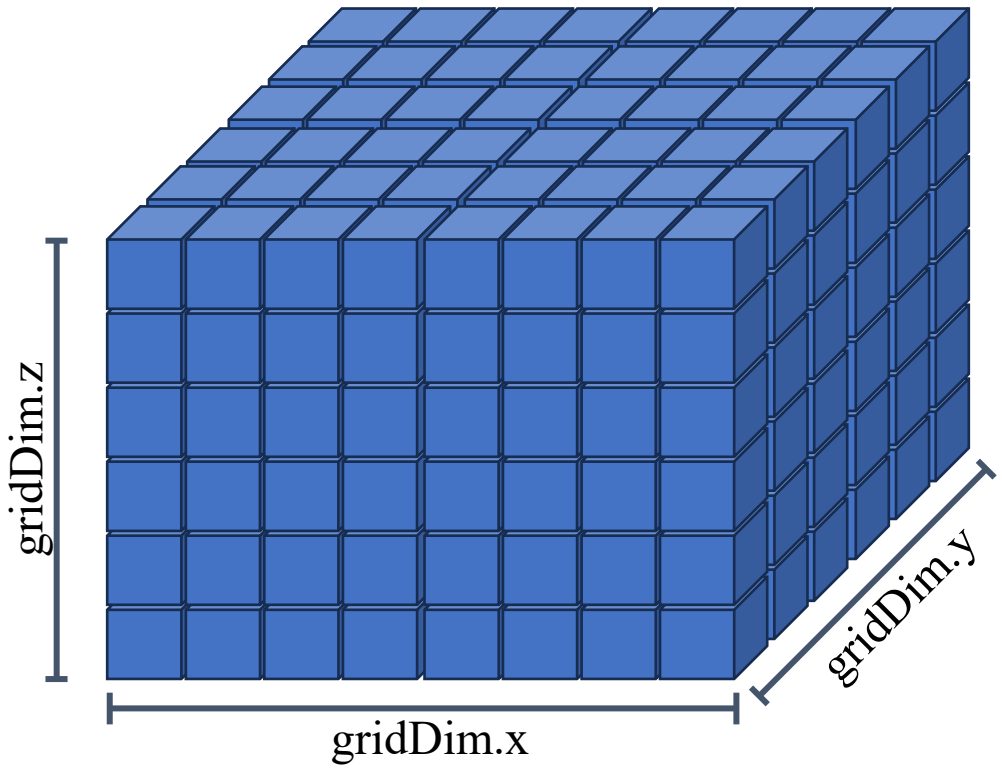

performWork<<<2, 4>>>()
4
4
blockDim.x


performWork<<<2, 4>>>()
0 1 2 3
0 1 2 3
threadIdx.x

网格与线程（二维和三维）

- gridDim.x
- gridDim.y
- gridDim.z

- blockIdx.x
- blockIdx.y
- blockIdx.z



- blockDim.x
- blockDim.y
- blockDim.z

- threadIdx.x
- threadIdx.y
- threadIdx.z


NVIDIA