NVIDIA | DEEP LEARNING INSTITUTE

加速计算基础 —— CUDA Python

网格跨度循环

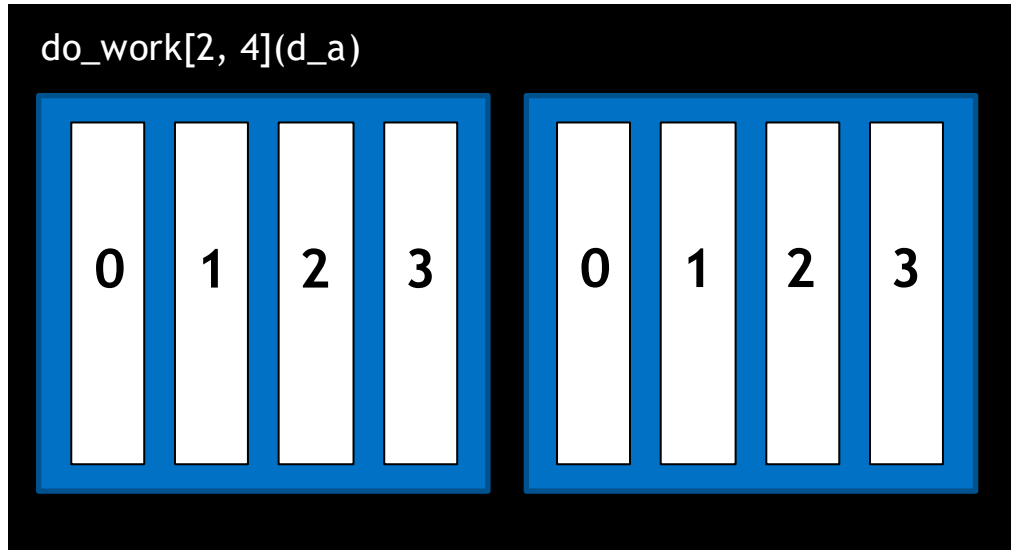
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

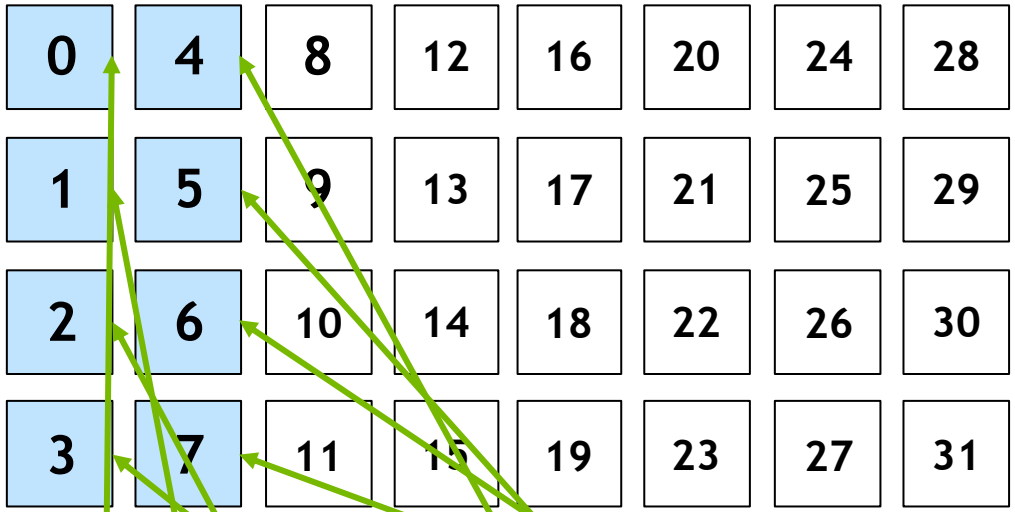
数据元素数量往往会大于网格中的线程数量

GPU



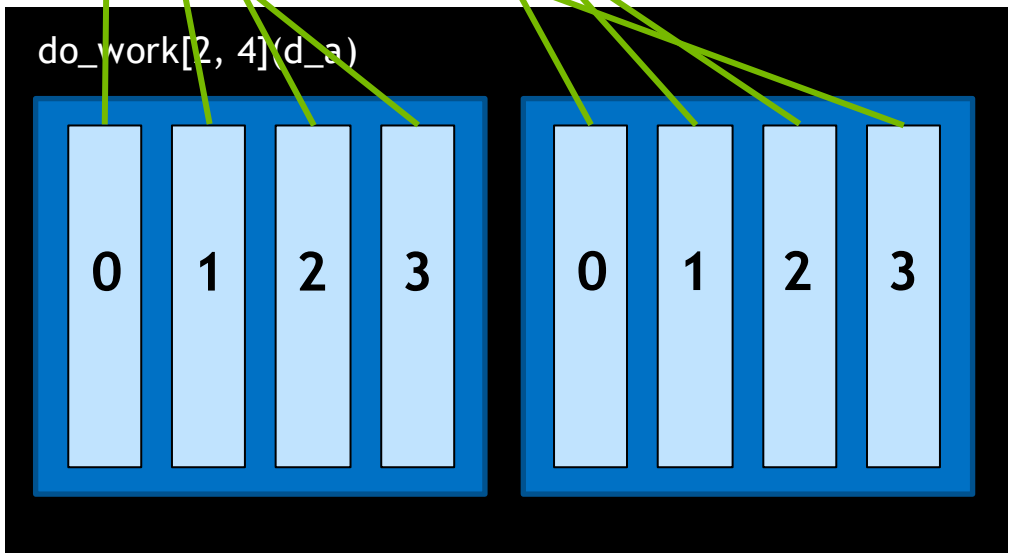
GPU

数据



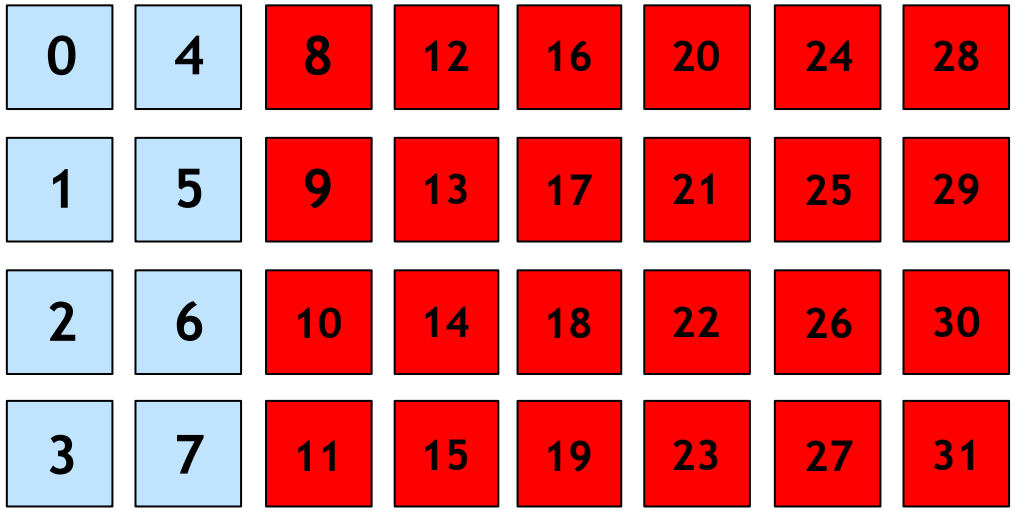
在此类情况下，线程无法只处理一个元素

GPU



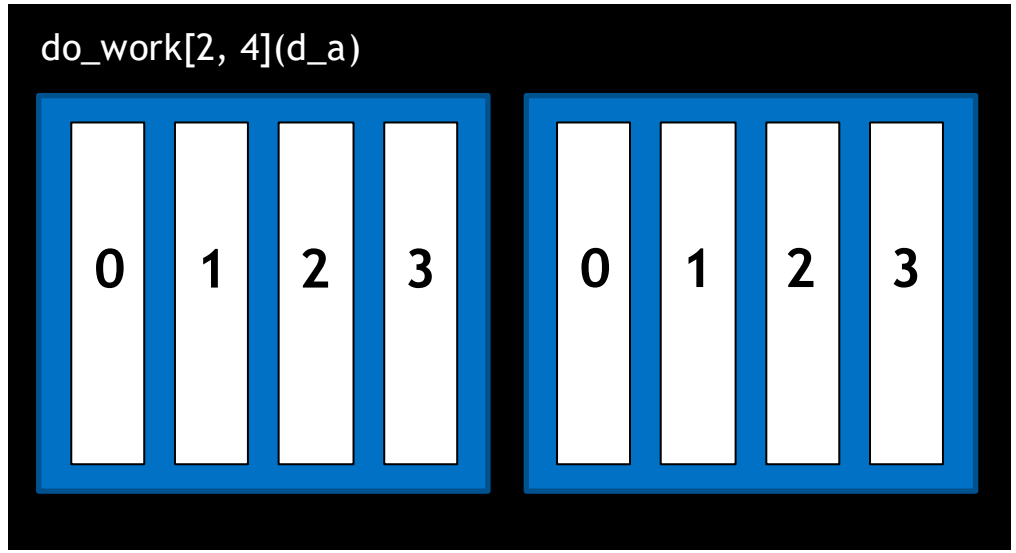
GPU

数据



...否则工作便无法完成

GPU



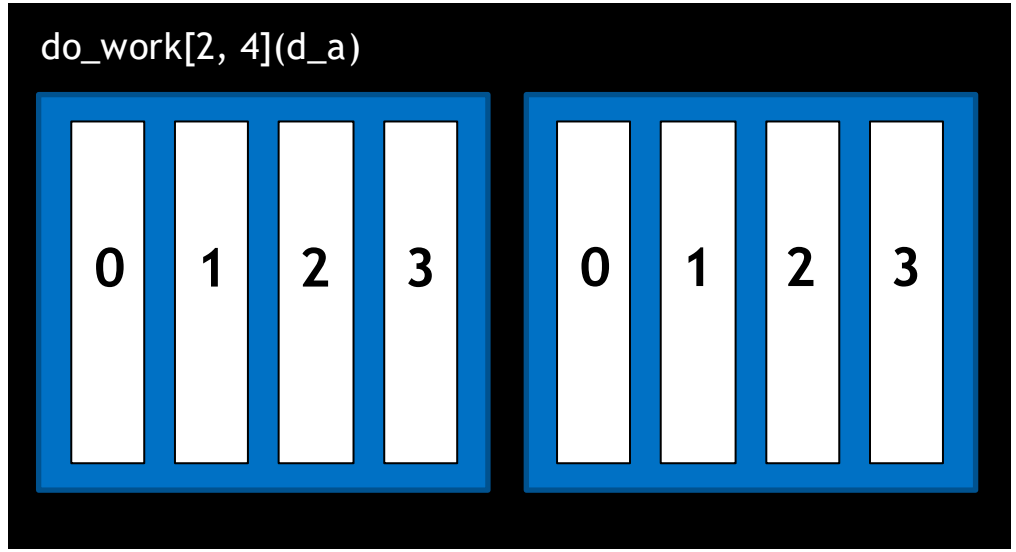
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

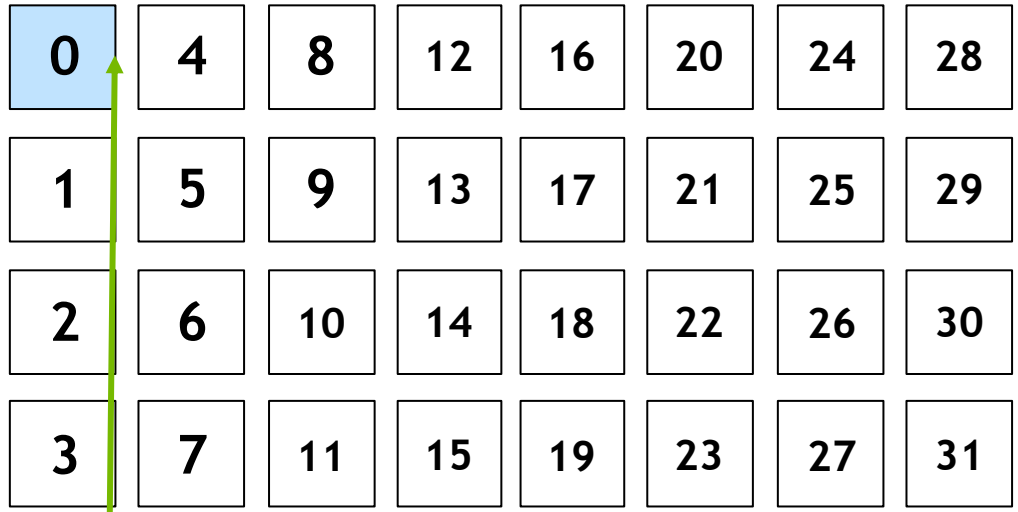
以编程方式解决此问题的方法之一是使用网格跨度循环

GPU



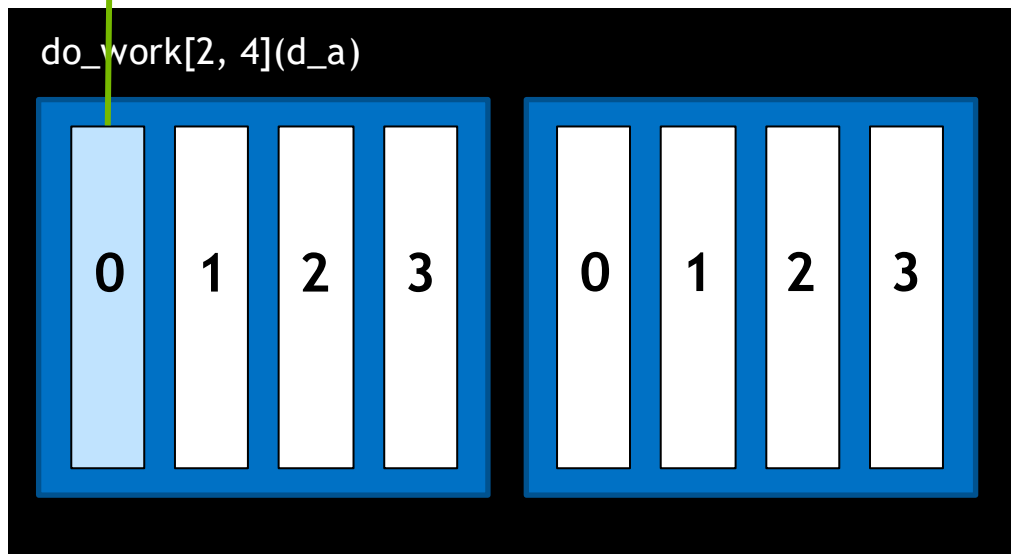
GPU

数据



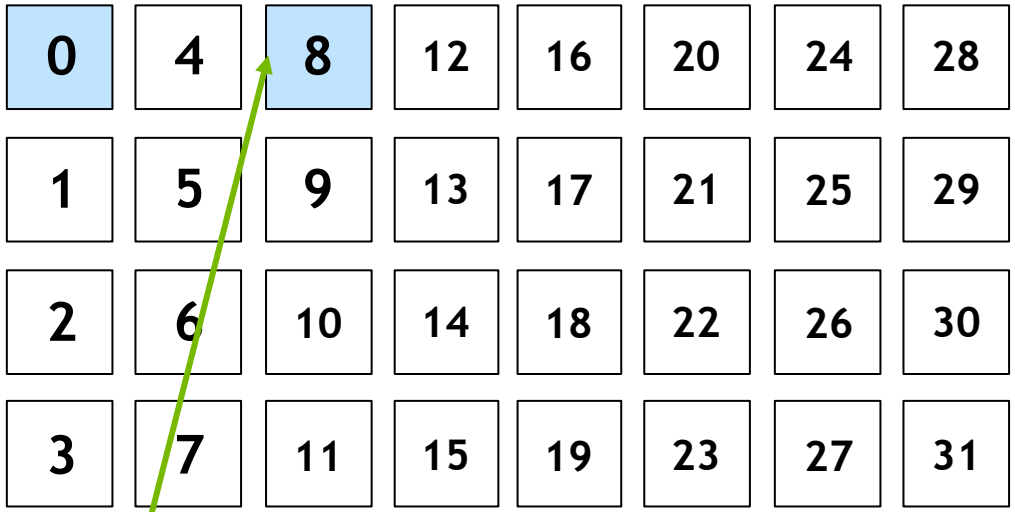
在网格跨度循环中，
线程的第一个元素依旧使用
cuda.grid() 计算得出

GPU



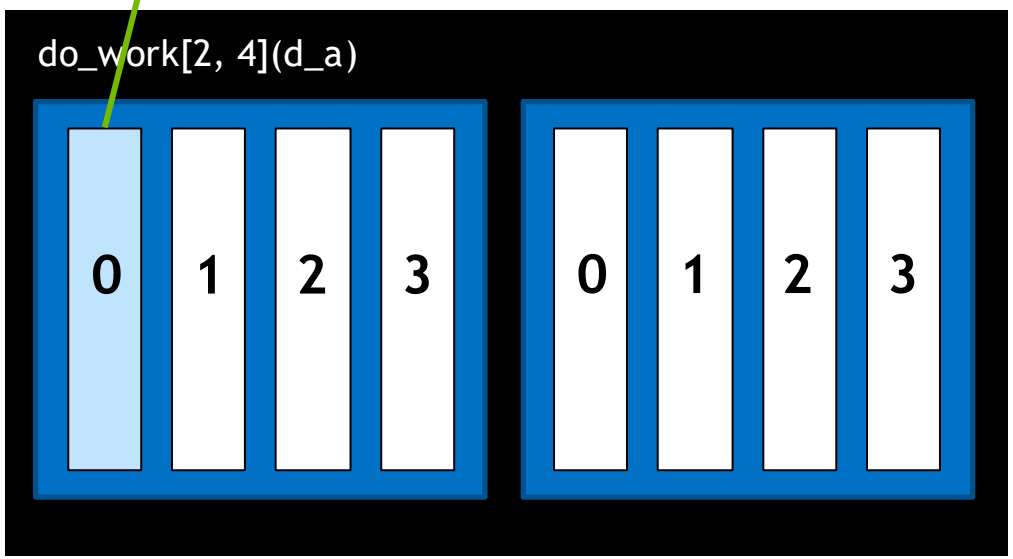
GPU

数据



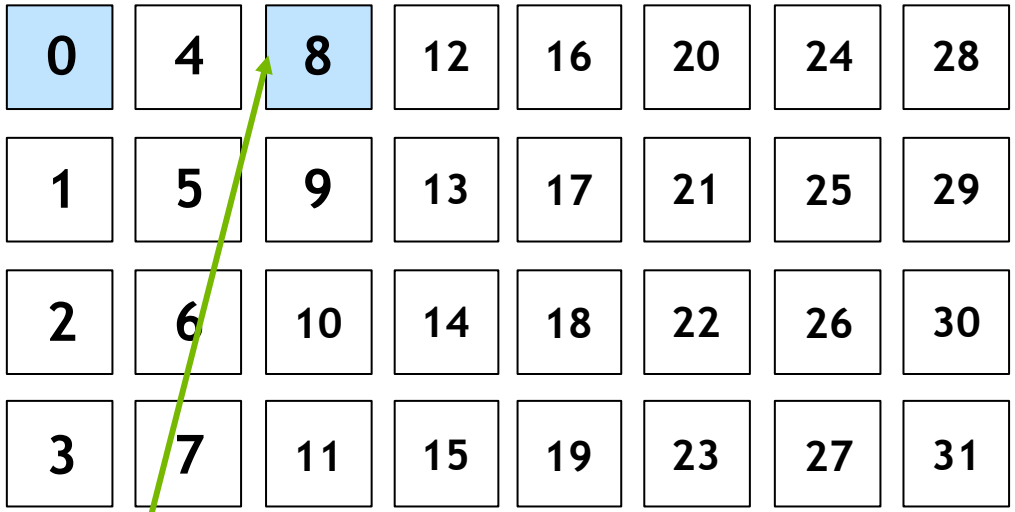
然后，线程会按网格中的线程总数 (**blockDim.x * gridDim.x**) **向前跃进**，在本例中线程总数为 **8**

GPU



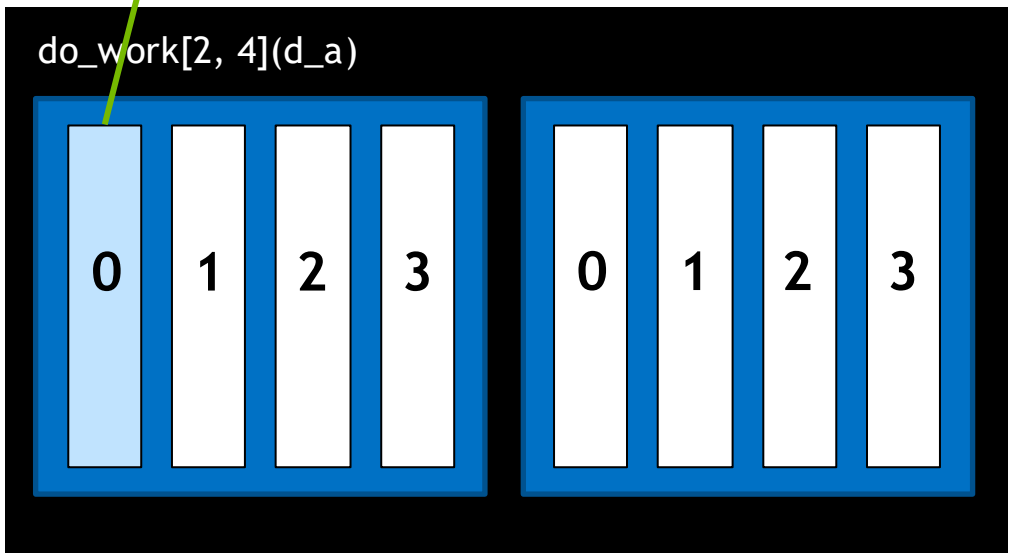
GPU

数据



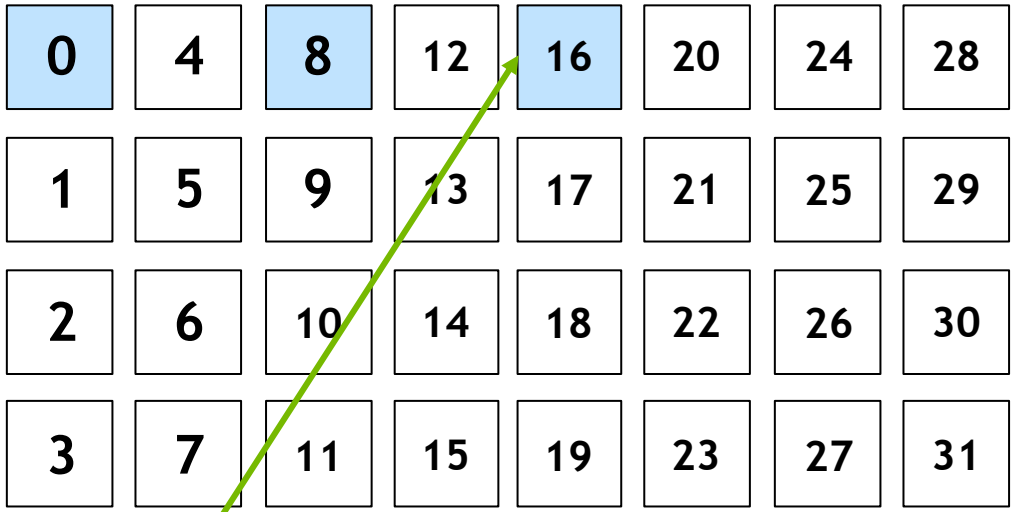
Numba 还为这一常用计算提供了另一个便捷函数 **cuda.gridsize()**，该函数可返回网格中的线程数量

GPU



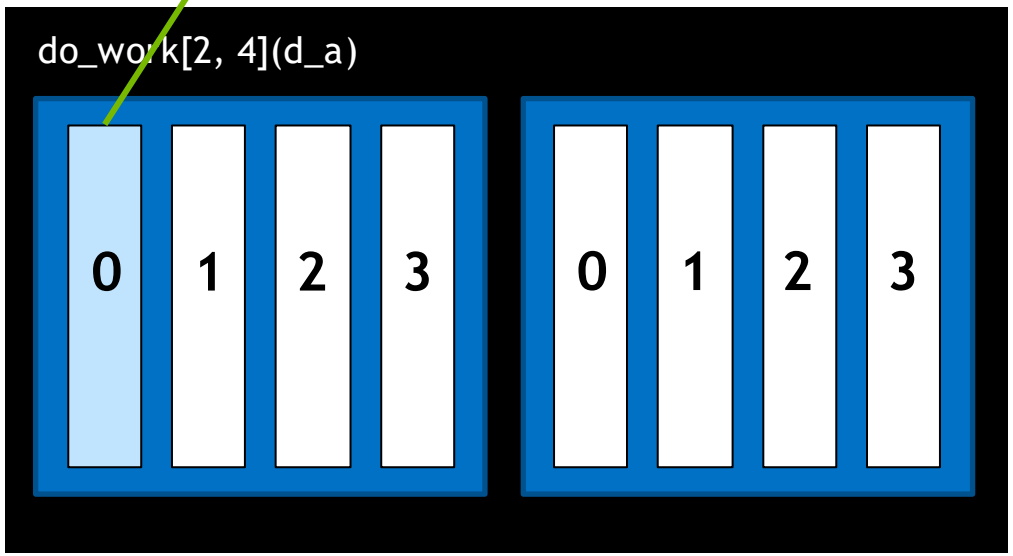
GPU

数据



线程会继续向前跃进，
直至其数据索引超出数据
元素的数量

GPU



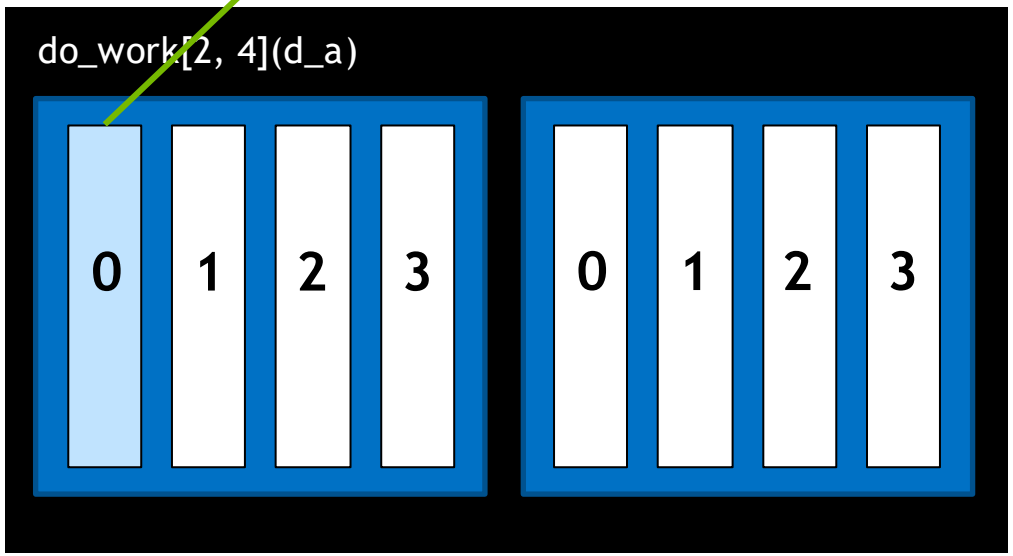
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

线程会继续向前迈进，
直至其数据索引超出数据
元素的数量

GPU



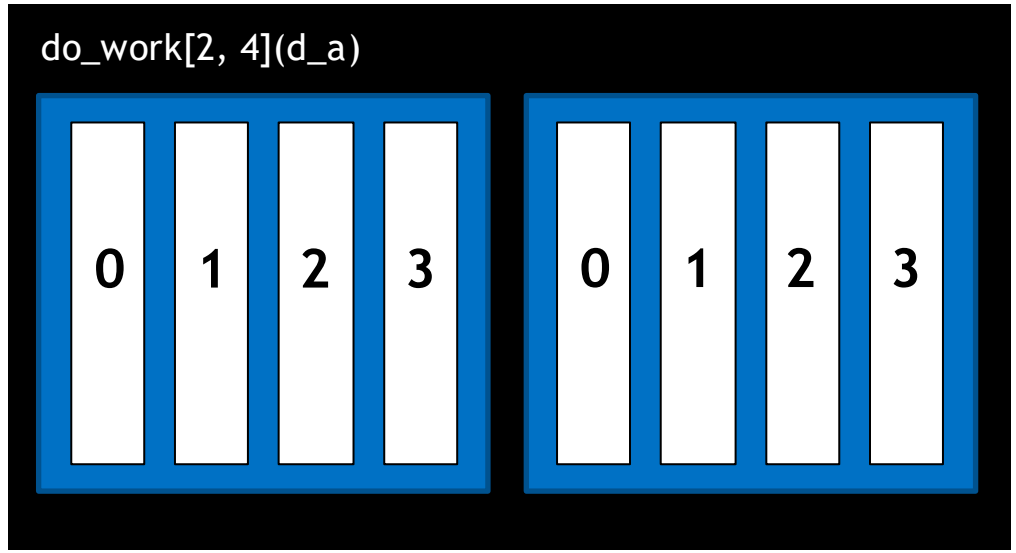
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

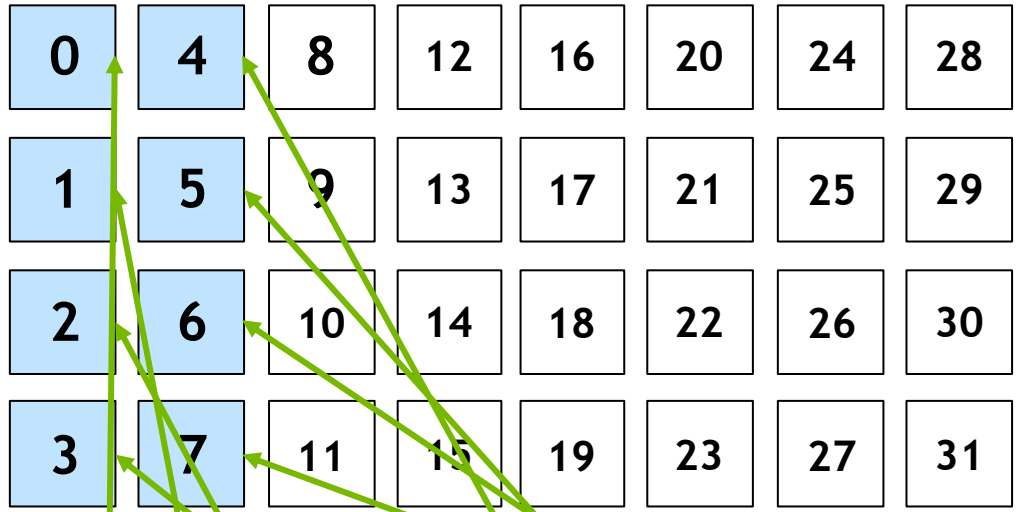
所有线程并行地执行网格跨度循环...

GPU



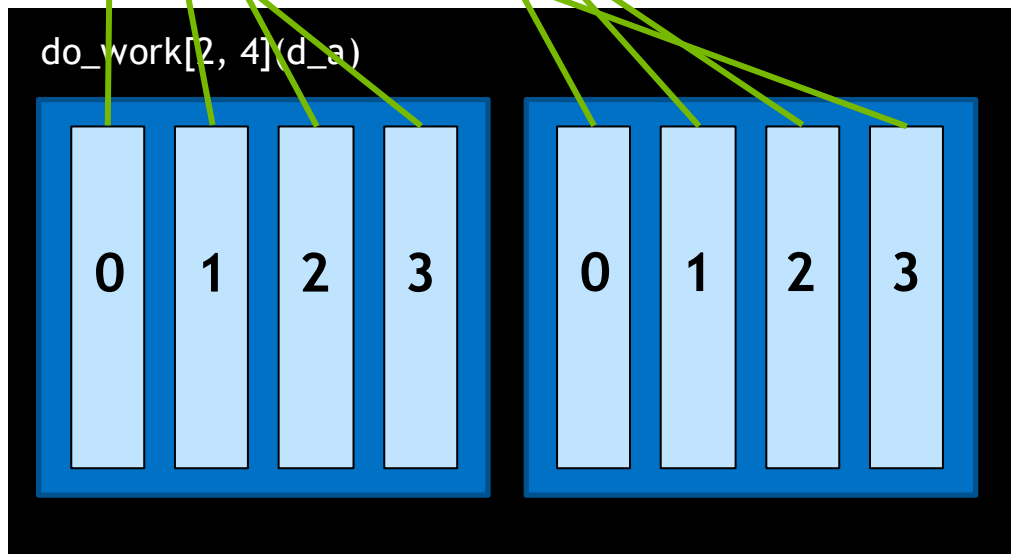
GPU

数据



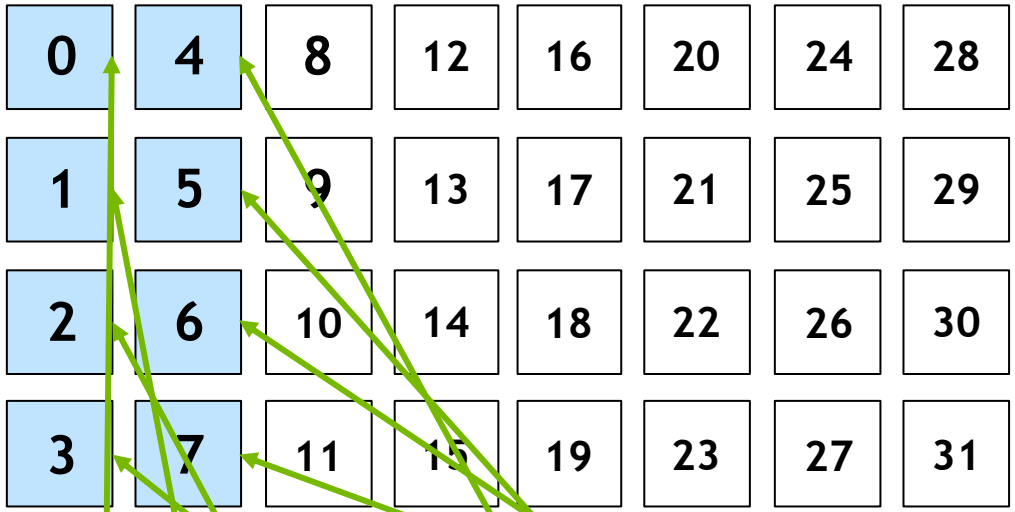
...所有元素便可涵盖在内

GPU



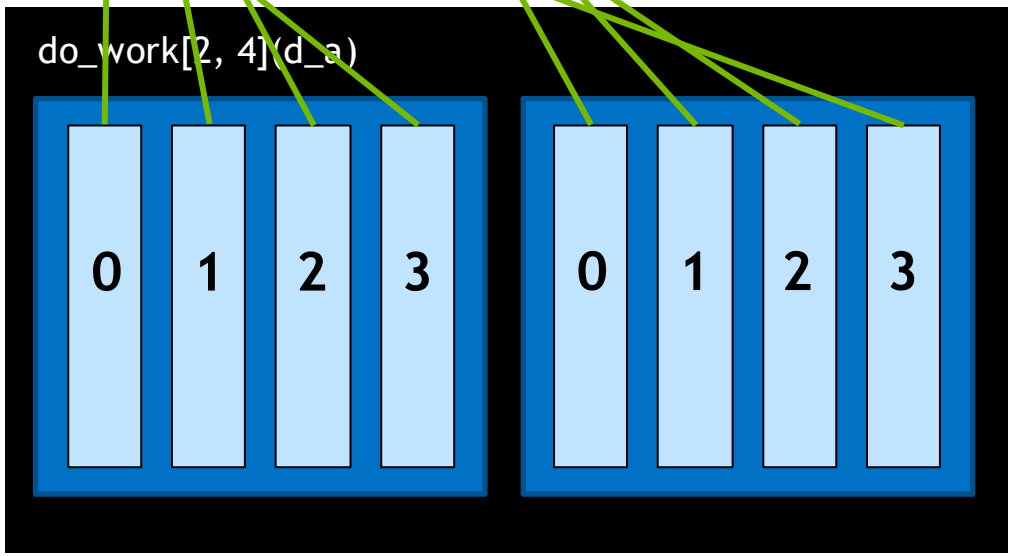
GPU

数据



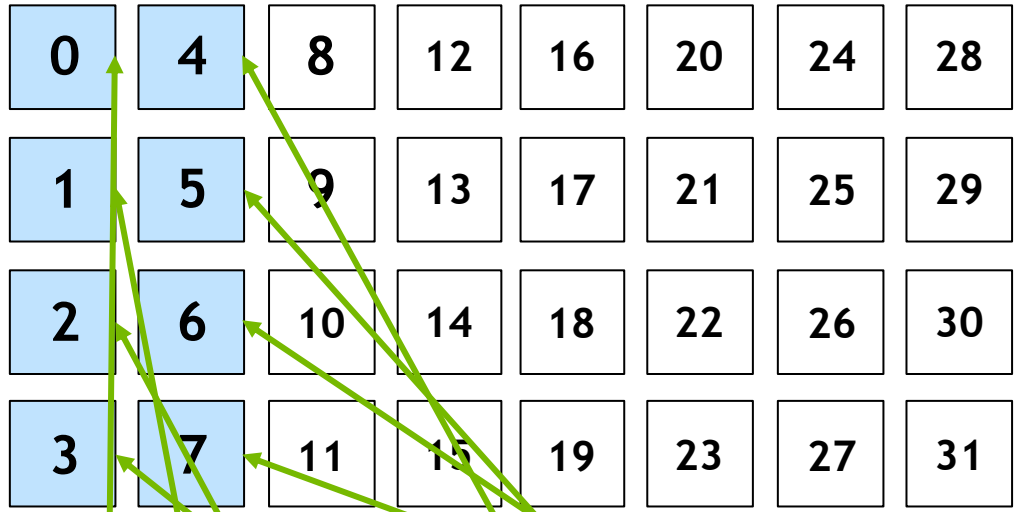
此外，设备还会将内存读写**合并**，**尽可能减少事务数量**，以获得更强性能...

GPU



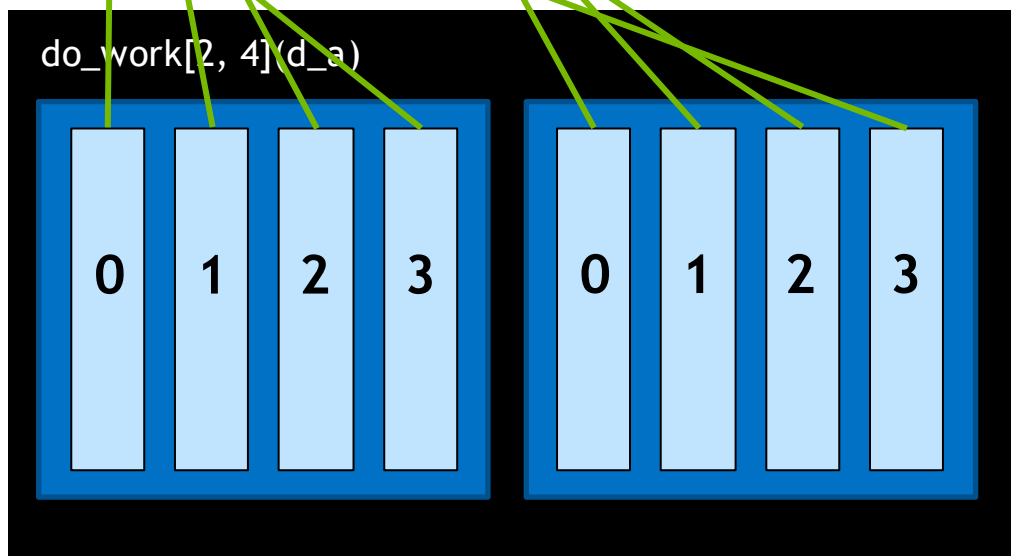
GPU

数据



网格跨度循环支持这种访存合并，因为线程在并行执行时会访问相邻的数据元素

GPU



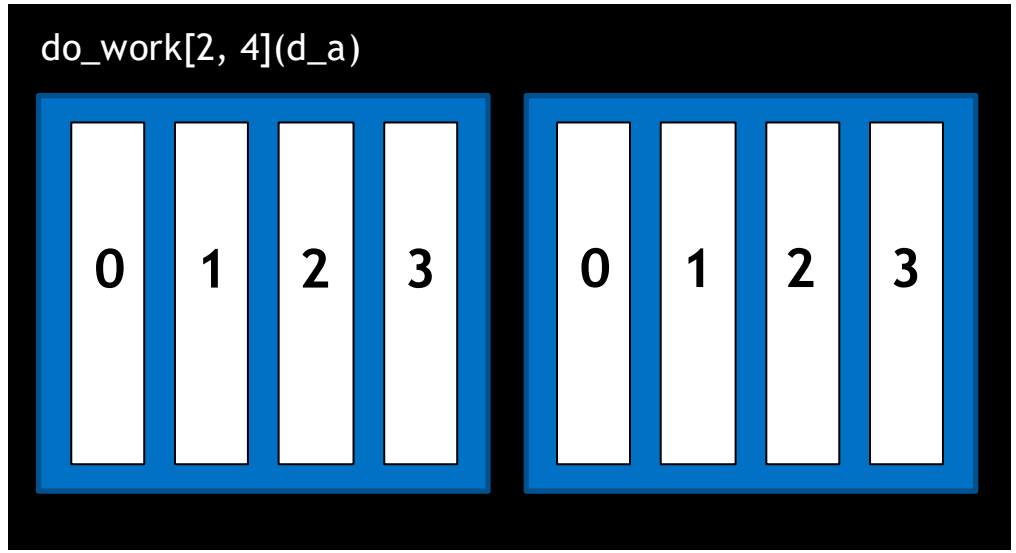
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

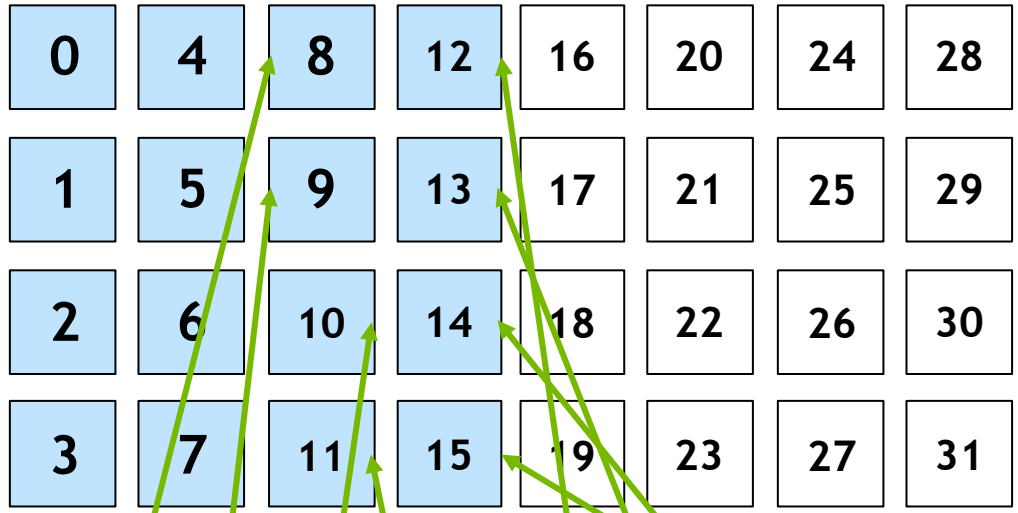
若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



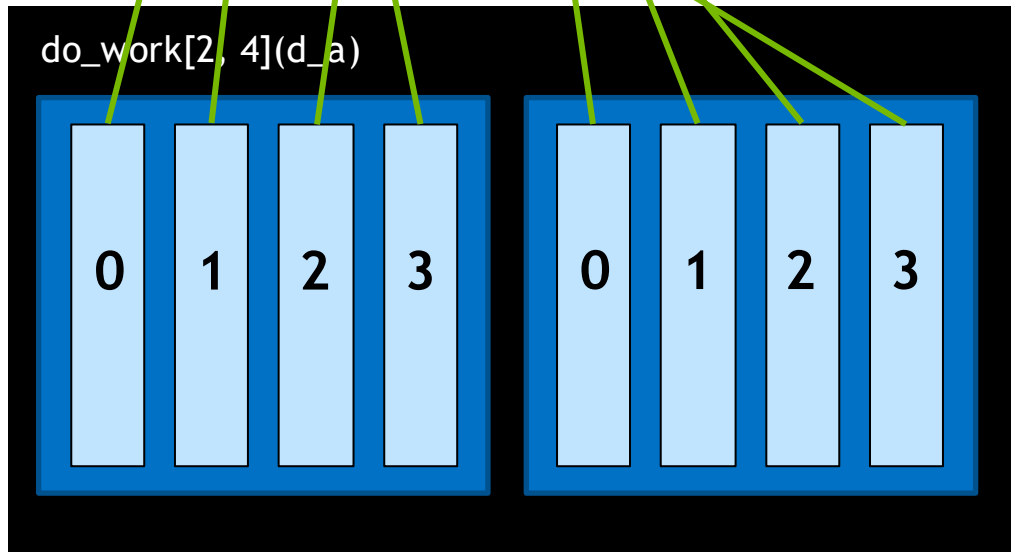
GPU

数据



若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



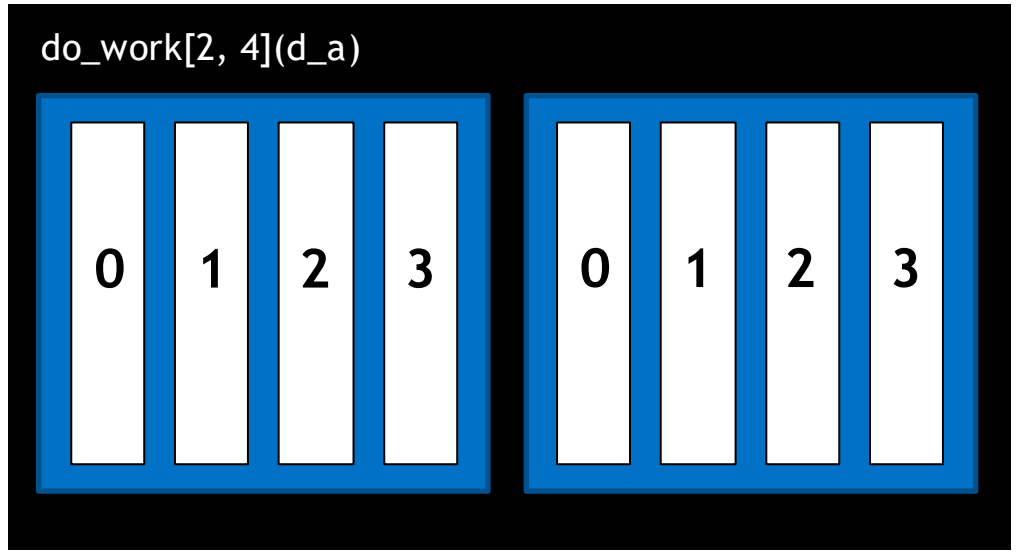
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

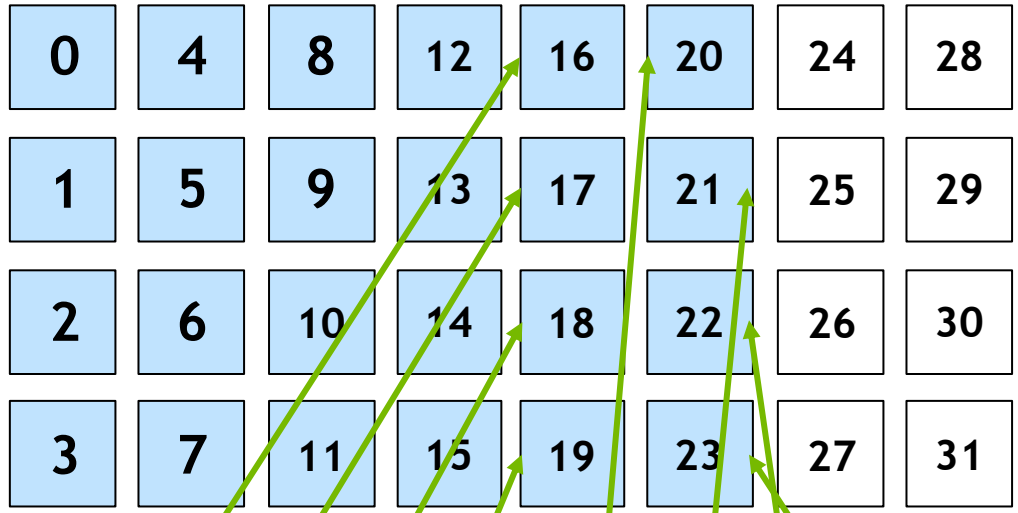
若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



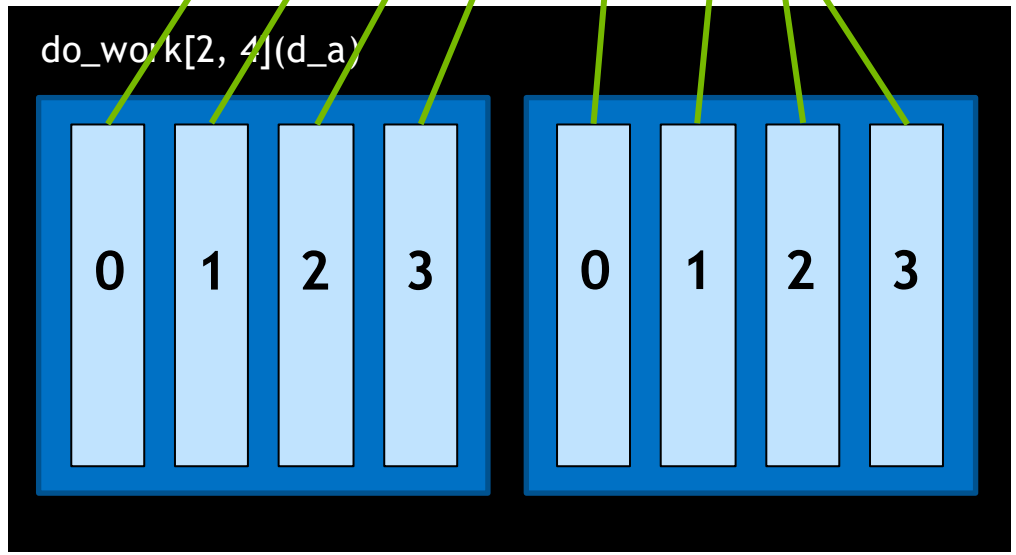
GPU

数据



若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



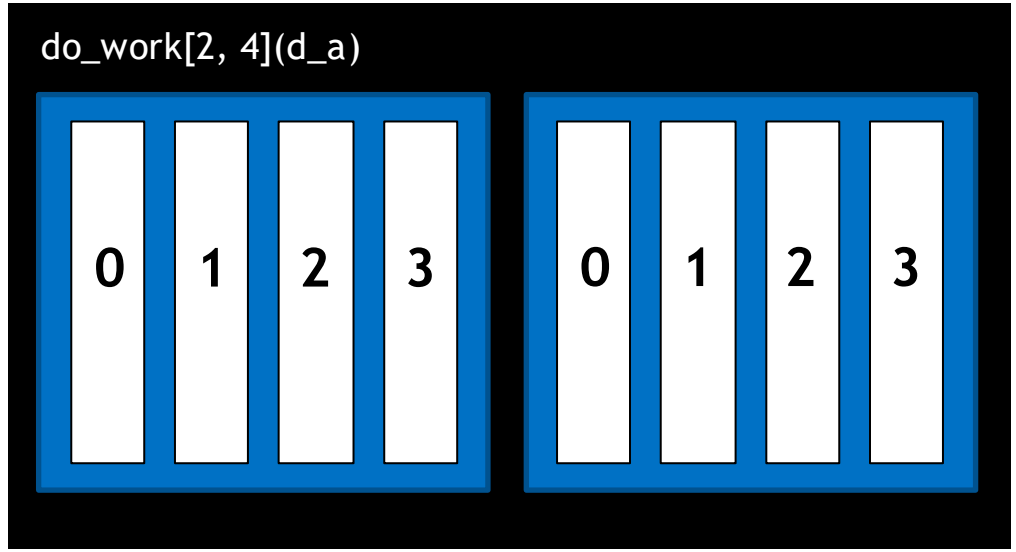
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

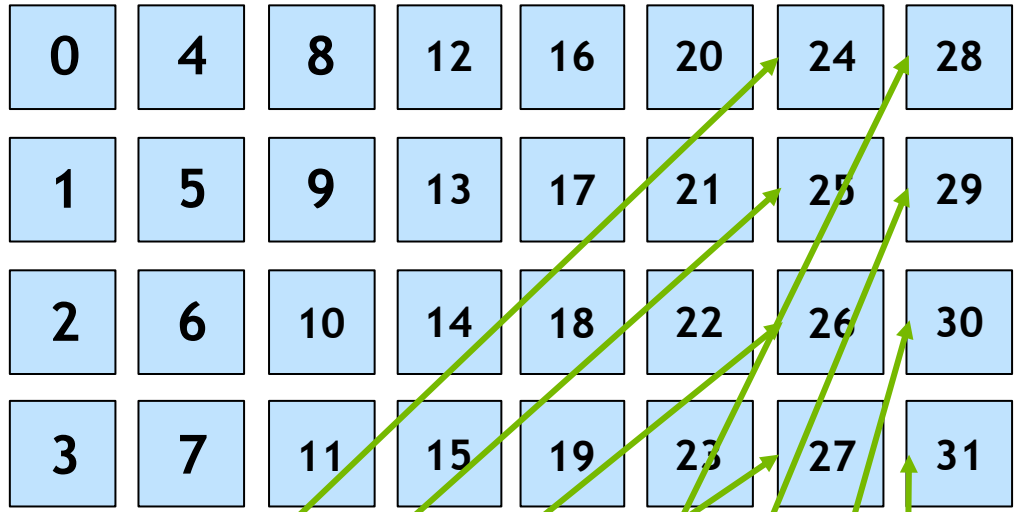
若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



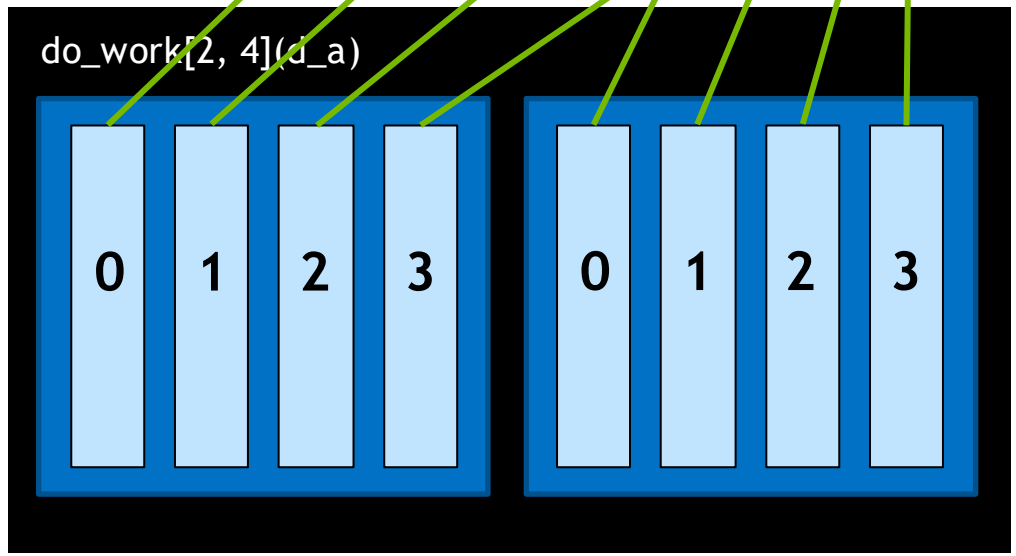
GPU

数据



若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



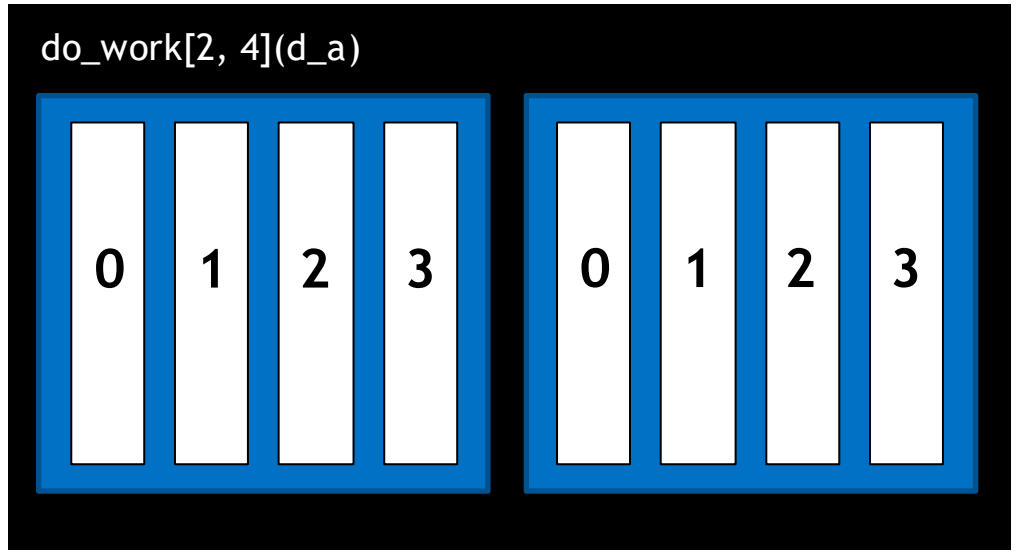
GPU

数据

0	4	8	12	16	20	24	28
1	5	9	13	17	21	25	29
2	6	10	14	18	22	26	30
3	7	11	15	19	23	27	31

若所有线程均按此种方式执行，所有元素便可覆盖在内，同时还可尽享访存合并所带来的性能优势

GPU



NVIDIA | DEEP LEARNING INSTITUTE

学习更多课程，请访问 www.nvidia.cn/DLI