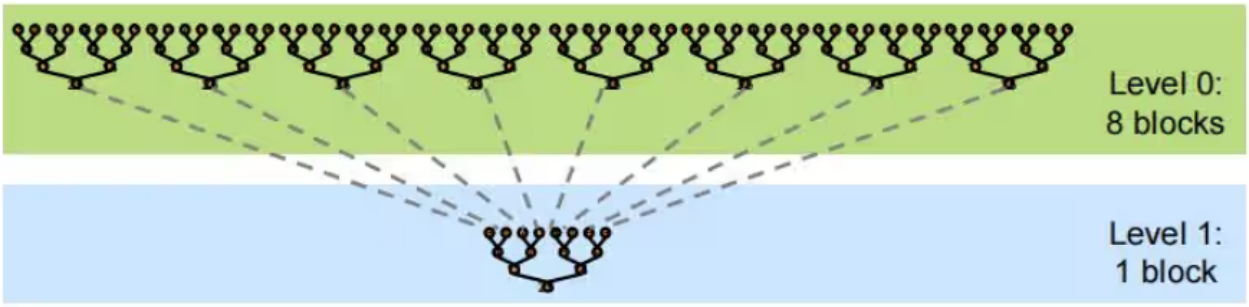
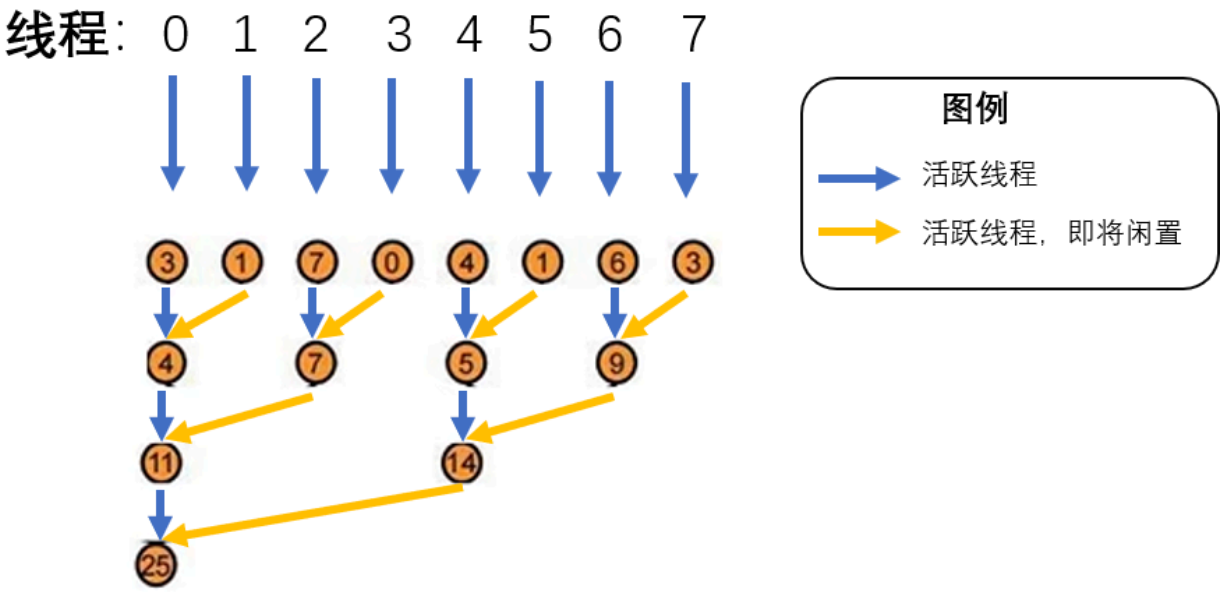
02 全局内存与共享内存

算法背景

如下图所示，`N`个元素两两做规约加法，分配`block_num`个CUDA block去操作。



在每一个`block`中，执行流程如下：



算法实现

1. Global 内存实现

```
/* 'reduce' kernel function */
__global__ void reduce(float *d_input, float *d_output) {

    float *input_begin = d_input + blockIdx.x * blockDim.x;    /* pointer to each
block beginning position */

    for (int i = 1; i < blockDim.x; i *= 2) {
        if (threadIdx.x % (i * 2) == 0) {
            input_begin[threadIdx.x] += input_begin[threadIdx.x + i];
        }
        __syncthreads();
    }

    if (threadIdx.x == 0) {
        d_output[blockIdx.x] = input_begin[0];   /* store result to global memory
*/
```

```
    }

}

/* 'reduce' kernel function */
__global__ void reduce_v2(float *d_input, float *d_output) {

    int index = blockDim.x * blockIdx.x + threadIdx.x; /* pointer to global positio
n */

    for (int i = 1; i < blockDim.x; i *= 2) {
        if (threadIdx.x % (i * 2) == 0) {
            d_input[index] += d_input[index + i];
        }
        __syncthreads();
    }

    if (threadIdx.x == 0) {
        d_output[blockIdx.x] = d_input[index];   /* store result to global memory
*/
    }

}
```

2. Shared 内存实现

```
__global__ void reduce(float *d_input, float *d_output) {

    __shared__ float shared_data[THREADS_PER_BLOCK];   /* create array in sh
ared memory */
    int index = blockDim.x * blockIdx.x + threadIdx.x; /* pointer to global positio
n */
    shared_data[threadIdx.x] = d_input[index];         /* load data : global memor
y → shared memory */
```

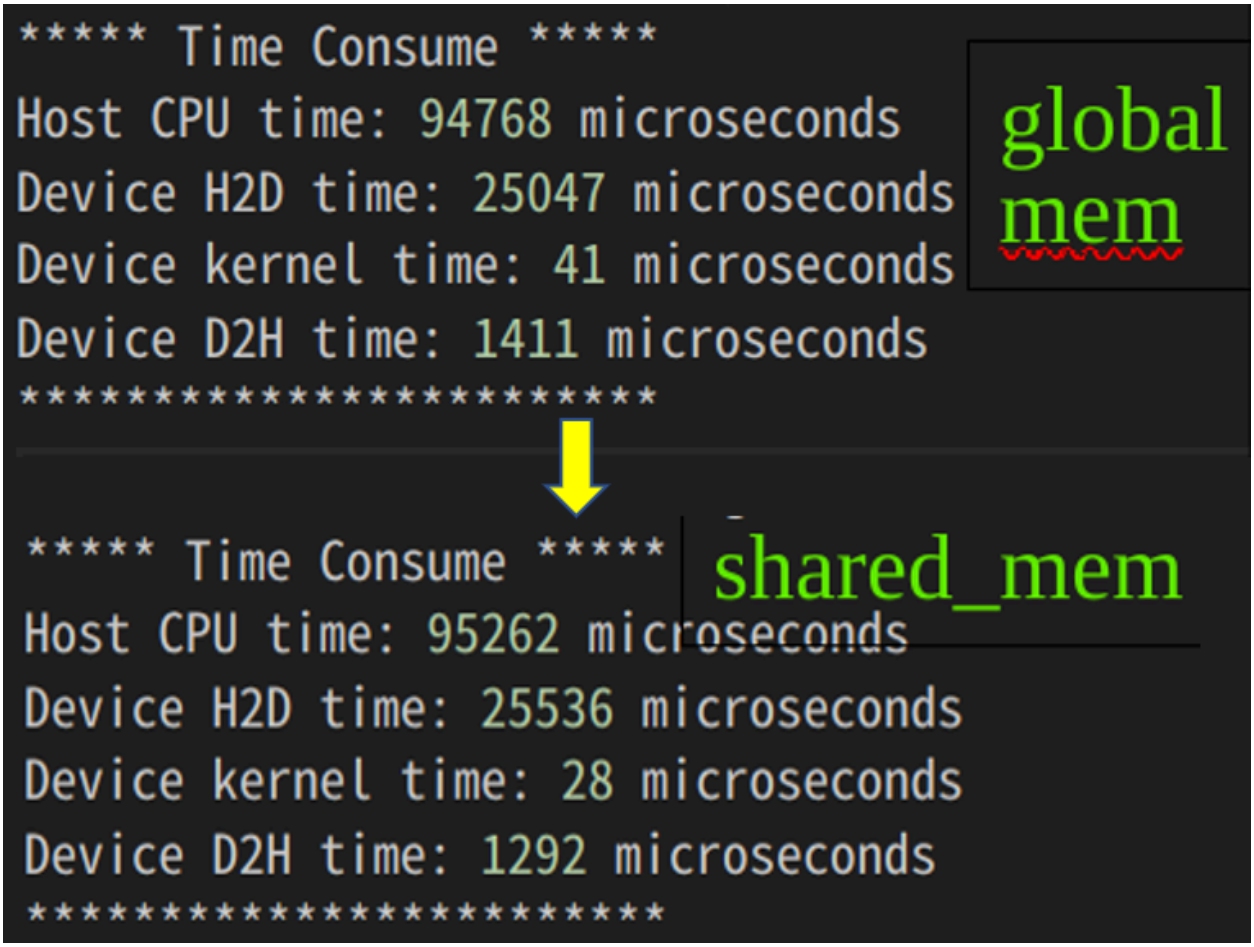
```
    __syncthreads();    /* ensure all threads have loaded data */

    for (int i = 1; i < blockDim.x; i *= 2) {
        if (threadIdx.x % (i * 2) == 0) {
            shared_data[threadIdx.x] += shared_data[threadIdx.x + i];
        }
        __syncthreads();
    }

    if (threadIdx.x == 0) {
        d_output[blockIdx.x] = shared_data[0];   /* store result to global memory
*/
    }

}
```

3. 运行结果



在硬件层面，Global 内存是DDR/HBM，而Shared内存是SRAM，所以Shared内存上的计算速度比Global 内存上的计算速度要快。

工程链接

https://github.com/JingliangGao/CudaRoad/tree/main/reduce_case