CSCI 325
Introduction to Parallel Systems and GPU Programming

Lecture 7
CUDA with C++

Dr. Talgat Turanbekuly

Table of contents

Example of passing defined type to kernel

```
define custom type
    typedef float orange;
use in caller function
    orange c = 77;
    orange *dev_c;
    cudaMalloc((void**)&dev_c, sizeof(orange));
    cudaMemcpy(dev_c, &c, sizeof(orange), cudaMemcpyHostToDevice);
and in kernel
    __global__ void testFunction(float *dev_a, float *dev_b, orange *dev_c)
```

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_2/main.cu

Example of custom defined function

```
#define customDefinedFunction(a, b)(a*b/(THREADS));
__device__ int deviceFunction(int a, int b)
{
  int c = a - b;
  int d = b - a;
  return max(c,d);
}
__global__ void globalFunction(float *dev_b)
{
  int index = threadIdx.x + blockIdx.x*blockDim.x;
  if(index == 0)
  {
    float a = customDefinedFunction(4, 16);
    printf("%.2f\n", a);
    int b = deviceFunction(4, 16);
    printf("%d\n", b);
    dev_b[index] = a + b;
  }
}
```

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_3/kernel.cu

Example of passing variables from host to kernel

```
__global__ void testFunction(float *dev_a, float *dev_b, float *dev_c, float dev_d)

float a[THREADS] = { 1, 2, 3, 4, 5 };
float *dev_a;
cudaMalloc((void**)&dev_a, THREADS*sizeof(float));
cudaMemcpy(dev_a, a, THREADS*sizeof(float), cudaMemcpyHostToDevice);

float b = 25;
float *dev_b;
cudaMalloc((void**)&dev_b, sizeof(float));
cudaMemcpy(dev_b, &b, sizeof(float), cudaMemcpyHostToDevice);

float *dev_c;
cudaMallocManaged(&dev_c, sizeof(float));

float d = 77;
testFunction<<<BLOCKS, THREADS>>>(dev_a, dev_b, dev_c, d);
```

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_4/main.cu

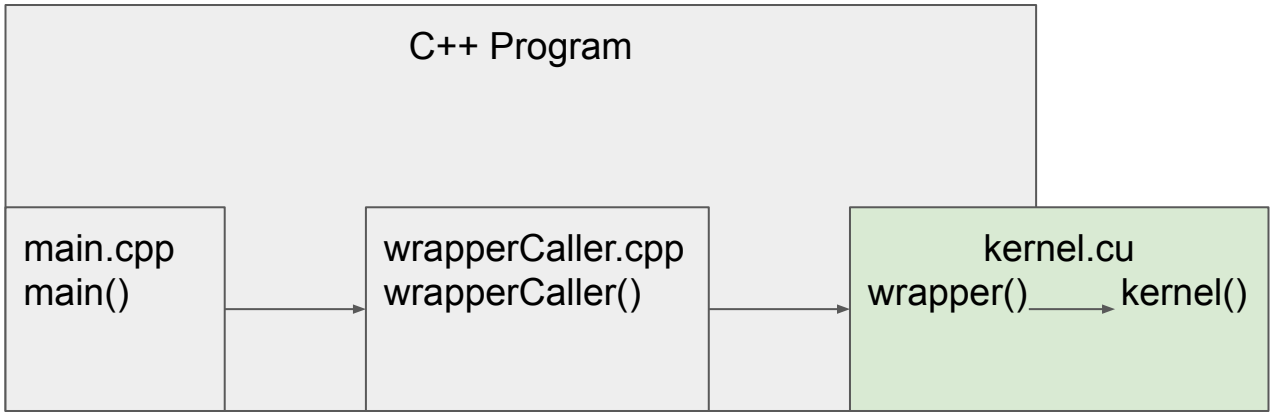
Example of calling device functions from kernel

```
__device__ int square(float numberInSquare)
{
  return numberInSquare * numberInSquare;
}
__device__ void cube(float numberInCube)
{
  numberInCube = square(numberInCube) * numberInCube;
  printf("numberInCube     = %.2f;\n", numberInCube);
}
__global__ void testFunction(float numberInGlobal)
{
  int thread = threadIdx.x;
  if(thread == 0)
  {
    printf("numberInGlobal 1 = %.2f;\n", numberInGlobal);
    numberInGlobal = square(numberInGlobal);
    printf("numberInGlobal 2 = %.2f;\n", numberInGlobal);
    cube(numberInGlobal);
    printf("numberInGlobal 3 = %.2f;\n", numberInGlobal);
  }
}
```

- device functions can return value
- they run within global function configurations

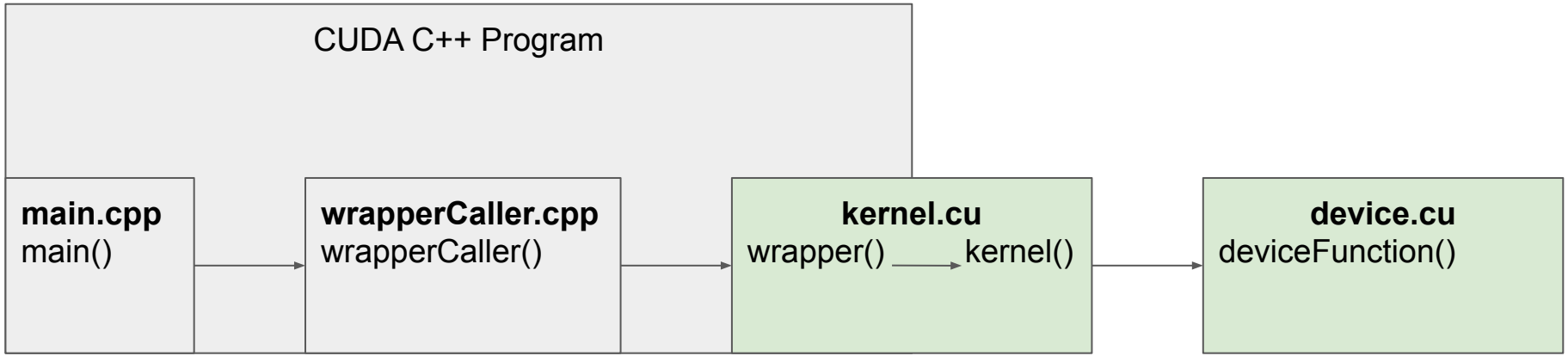
https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_5/main.cu

Example of calling CUDA function from C++ file



https://github.com/manglayev/ItPSaGPUP/tree/main/Lecture_7/Example_1

Example of calling device functions in other file from kernel



https://github.com/manglayev/ItPSaGPUP/tree/main/Lecture_7/Example_6

Example of using enums in kernel

```
enum enumOrder {h1, h2, h3, h4};

__global__ void globalFunction(int a, int *b, enumOrder order)
{
  int thread = threadIdx.x + blockIdx.x*blockDim.x;
  if(thread < THREADS)
  {
    switch(order)
    {
      case h1:
        b[0] = a+a;
        break;
      case h2:
        b[0] = a-a;
        break;
      case h3:
        b[0] = a*a;
        break;
      default:
        b[0] = a/a;
        break;
    }
  }
}

globalFunction<<<BLOCKS, THREADS>>>(a, b, h1);
```

https://github.com/manglayev/ItPSaGPUP/tree/main/Lecture_7/Example_7

Example of defining the same function to run on the host and on device

```
__host__ __device__ int customFunction(int a, int *b)
{
  return a+a;
}
```

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_8/main.cu

Example of C++ function template

Allows to call and run functions with different parameters (use as data type)

```
template<typename T> T signum(T x)
{
 if(x > 0)
   return 1;
 else if(x < 0)
   return -1;
 else
   return 0;
}
```

typename T can be int, float etc.

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_9/main.cu

Example of CUDA C++ function template

```
__host__ __device__ to specify the function call

template<typename T> __host__ __device__ T signum(T x)
{
 if(x > 0)
   return 1;
 else if(x < 0)
   return -1;
 else
   return 0;
}
```

https://github.com/manglayev/ItPSaGPUP/tree/main/Lecture_7/Example_10

Example of CUDA C++ class template

```
#define CUDA_HOSTDEV __host__ __device__

template <typename T> class Array
{
 private:
   T* ptr;
   int size;

 public:
   CUDA_HOSTDEV Array(T arr[], int s);
   CUDA_HOSTDEV void print();
};
```

https://github.com/manglayev/ItPSaGPUP/blob/main/Lecture_7/Example_11/main.cu