CUDA C

Enjoy this cheat sheet at its fullest within Dash, the macOS documentation browser.

Function Type Qualifiers

Function type qualifiers are used in function declarations and definitions.

They specify where the functions can be called from and where they can be executed.

Host is the CPU side, and device is the GPU side.

`__device__`

Executed on the device. Callable from the device only.

`__global__`

Executed on the device. Callable from the host or from the device for devices of compute capability 3.x or higher. Must have void return type.

`__host__`

Executed on the host. Callable from the host only (equivalent to declaring the function without any qualifiers).

Built-in Vector Types

Types

`charX`, `ucharX`, `shortX`, `intX`, `uintX`, `longX`, `ulongX`, `floatX`, where `X = 1, 2, 3, or 4`.

`doubleX`, `longlongX`, `ulonglongX`, where `X = 1, or 2`.

Note: `dim3` is a `uint3` with default components initalized to 1.

Constructor Function

`make_<type name>` constructs the built-in vector type with type specified by replacing `<type name>` with one of the types above.

Component Access

The 1st, 2nd, 3rd, and 4th components are accessible through the fields `x`, `y`, `z`, and `w`.

Example

`int4 intVec = make_int4(0, 42, 3, 5)` creates an `int4` vector typed variable named `intVec` with the given `int` elements. `intVec.z` accesses its third element, `3`.

Built-in Variables

Inside functions executed on the device (GPU), grid and block dimensions, and block and thread indices can be accessed using built-in variables listed below.

`gridDim`

Dimensions of the grid (`dim3`).

`blockIdx`

Block index within the grid (`uint3`).

`blockDim`

Dimensions of the block (`dim3`).

`threadIdx`

Thread index within the block (`uint3`).

`warpSize`

Warp size in threads (`int`).

Device Memory Management

Allocating memory

```
cudaError_t cudaMalloc(void **devPtr, size_t size)
```

Allocates `size` bytes of linear memory on the device and points `devPtr` to the allocated memory.

Freeing memory

```
cudaError_t cudaFree(void *devPtr)
```

Frees the memory space pointed to by `devPtr`.

Transferring data

```
cudaError_t cudaMemcpy(void *dst, const void *src, size_t count, cudaMemcpyKind kind)
```

Copies `count` bytes of data from the memory area pointed to by `src` to the memory area pointed to by `dst`. The direction of copy is specified by `kind`, and is one of `cudaMemcpyHostToHost`, `cudaMemcpyHostToDevice`, `cudaMemcpyDeviceToHost`, or `cudaMemcpyDeviceToDevice`.

Kernel Launch

A kernel function declared as `__global__ void Func(float *parameter)` can be called without the optional arguments as `Func<<<numBlocks, threadsPerBlock>>>(parameter)` or with the optional arguments as `Func<<<numBlocks, threadsPerBlock, Ns, S>>>(parameter)`.

`numBlocks` is of type `dim3` and specifies the number of blocks,

`threadsPerBlock` is of type `dim3` and specifies the number of threads per block,

`Ns` is of type `size_t` and specifies bytes in shared memory (optional),

`S` is of type `cudaStream_t` and specifies associated stream (optional).

Notes

- Based on CUDA C Programming Guide and CUDA Runtime API.

You can modify and improve this cheat sheet here