

CUDA C Quick Reference

Kernels

kernel <<< dim3 Dg, dim3 Db, size_t Ns, cudaStream_t S >>> (arguments);

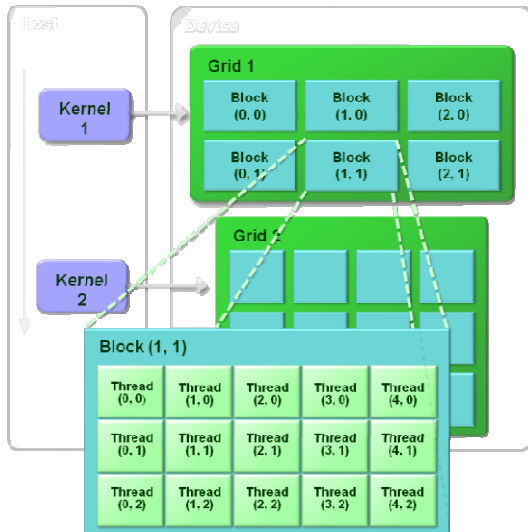
Dg.x*Dg.y = number of blocks, Dg.z = 1.

Db.x*Db.y*Db.z = number threads per block.

Ns = dynamically allocated shared memory, optional, default=0.

S = associated stream, optional, default=0.

Thread Hierarchy



Memory Hierarchy

Memory	Location	Cached	Access	Scope	Lifetime
Register	On-chip	N/A	R/W	Thread	Thread
Local	Off-chip	No	R/W	Thread	Thread
Shared	On-chip	N/A	R/W	Block	Block
Global	Off-chip	No	R/W	Global	Application
Constant	Off-chip	Yes	R	Global	Application
Texture	Off-chip	Yes	R	Global	Application

Device Memory

Linear Memory

cudaMalloc(void ** devptr, size_t size)

cudaFree(void * dptr)

cudaMemcpy(void *dst, const void *src, size_t size, enum cudaMemcpyKind kind)

kind = cudaMemcpyHostToHost or

cudaMemcpyHostToDevice or

cudaMemcpyDeviceToHost or

cudaMemcpyDeviceToDevice

CUDA Arrays

See Programming Guide for description of CUDA arrays and texture references.

Page-locked Host Memory

cudaMallocHost(void ** ptr, size_t size)

cudaFreeHost(void * ptr)

Shared Memory

Static allocation

```
__shared__ int a[128]
```

Dynamic allocation at kernel launch

```
extern __shared__ float b[]
```

Error Handling

cudaError_t cudaGetLastError(void)

```
const char * cudaGetErrorString( cudaError_t error )
```

CUDA Compilation

nvcc flags file.cu

A few common flags

-o output file name

-g host debugging information

-G device debugging

-deviceemu emulate on host

-use_fast_math use fast math library

-arch compile for specific GPU architecture

-X pass option to host compiler

#pragma unroll n unroll loop n times.

Language Extensions

Function Qualifiers

`__global__` call host, execute device.
`__device__` call device, execute device.
`__host__` call host, execute host (default).
`__noinline__` if possible, do not inline

`__host__` and `__device__` may be combined to generate code for both host and device.

Variable Qualifiers

`__device__` variable on device
`__constant__` variable in constant memory
`__shared__` variable in shared memory

Vector Types

`[u]char1`, `[u]char2`, `[u]char3`, `[u]char4`
`[u]short1`, `[u]short2`, `[u]short3`, `[u]short4`
`[u]int1`, `[u]int2`, `[u]int3`, `[u]int4`
`[u]long1`, `[u]long2`, `[u]long3`, `[u]long4`
`longlong1`, `longlong2`
`float1`, `float2`, `float3`, `float4`
`double1`, `double2`

Execution configuration

`kernel <<< dim3 Dg, dim3 Db, size_t Ns, cudaStream_t S >>> (arguments)`

Grids are 1D or 2D so `Dg.z = 1` always
`Ns` optional, default 0
`S` optional, default 0

Built-in Variables

`dim3 gridDim` size of grid (1D, 2D).
`dim3 blockDim` size of block (1D, 2D, 3D).
`dim3 blockIdx` location in grid.
`dim3 threadIdx` location in block.
`int warpSize` threads in warp.

Memory Fence Functions

`__threadfence()`, `__threadfence_block()`

Synchronisation Function

`__syncthreads()`

Fast Mathematical Functions

`__fdivdef(x,y)`, `__sinf(x)`, `__cosf(x)`, `__tanf(x)`,
`__sincosf(x,sinp,cosp)`, `__logf(x)`,
`__log2f(x)`, `__log10f(x)`, `__expf(x)`, `__exp10f(x)`,
`__powf(x,y)`

Texture Functions

`tex1Dfetch()`, `tex1D()`, `tex2D()`, `tex3D()`

Timing

`clock_t clock(void)`

Atomic Operations

`atomicAdd()`, `atomicSub()`, `atomicExch()`,
`atomicMin()`, `atomicMax()`, `atomicInc()`,
`atomicDec()`, `atomicCAS()`, `atomicAnd()`,
`atomicOr()`, `atomicXor()`.

Warp Voting Functions

`int __all(int predicate)`
`int __any(int predicate)`