Parallel Computing Workshop

Ms. Kranti Ingale

Prime Minister's Research Fellows (**PMRF**) Scholar

CSE, IIT Madras

Roadmap

- Introduction to GPU
- CUDA Program Flow and CPU-GPU Communication
- Thread organization (Grids, Blocks, Threads, 1D/2D)
- CUDA Memory Model
- CUDA Functions
- CUDA Thrust

Roadmap

- Introduction to GPU
- CUDA Program Flow and CPU-GPU Communication
- Thread organization (Grids, Blocks, Threads, 1D/2D)
- CUDA Memory Model
- CUDA Functions
- CUDA Thrust

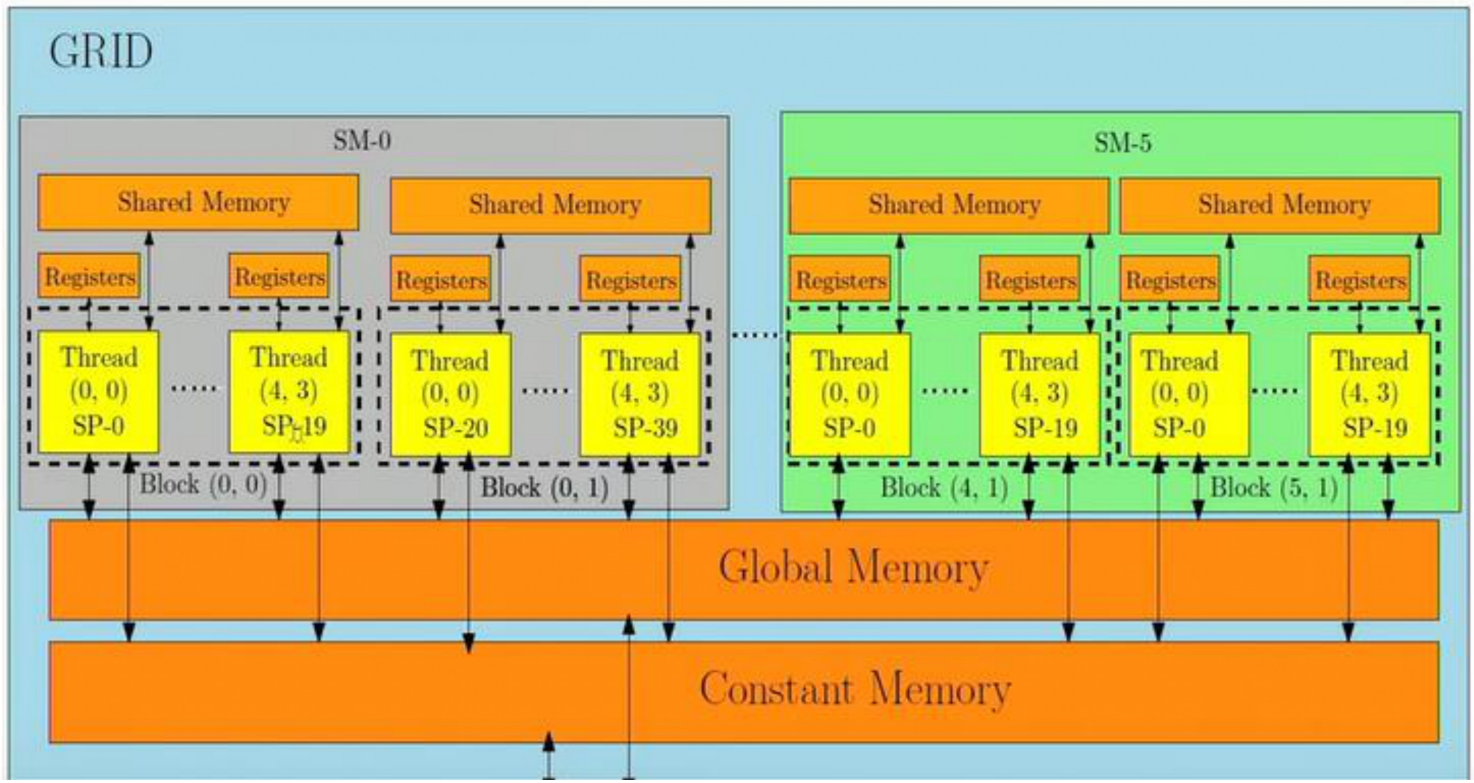
Recap

- Grid < 6,2,1> Block = <5,4,1>
- Number of SM's = 6
- SP's per SM = 40
- Number of thread block = 6 * 2 = 12

Recap

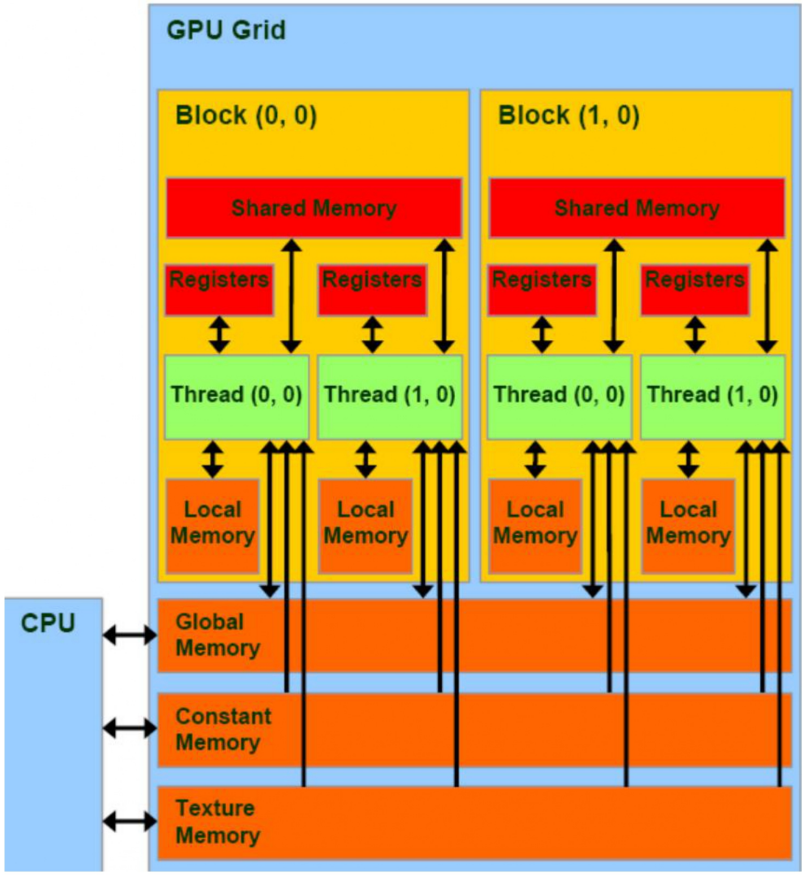
- Grid < 6,2,1> Block = <5,4,1>
- Number of SM's = 6
- SP's per SM = 40
- Number of thread block = 6 * 2 = 12
- Threads per thread_block = 5 * 4 = 20
- Two thread block are mapped to one SM

Recap - Thread mapping to Hardware



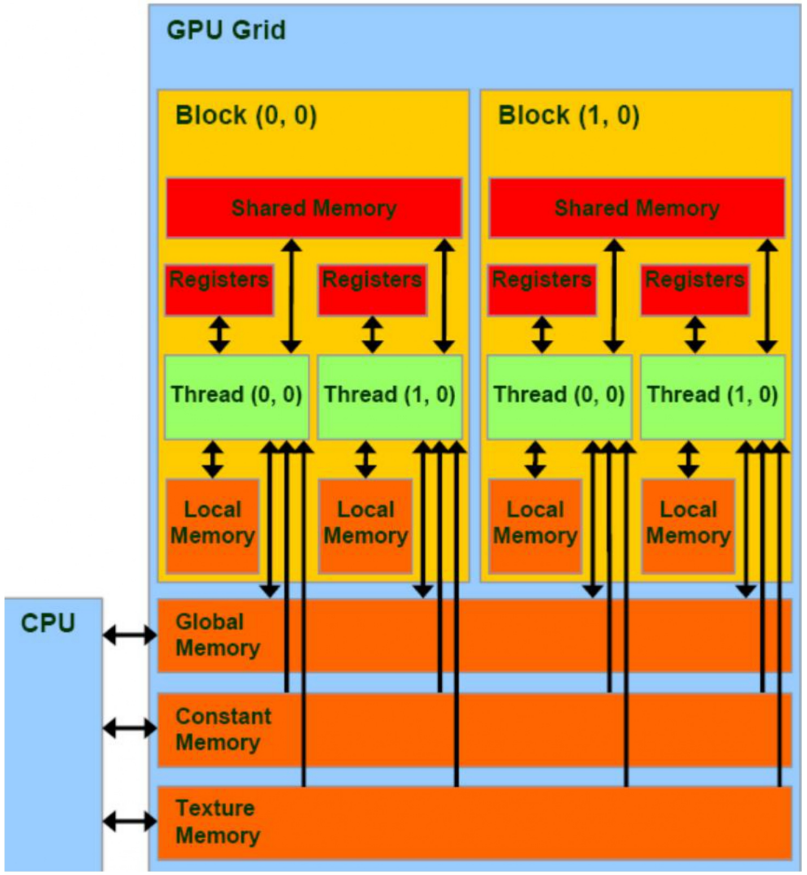
CUDA Memory Model Overview

- Global Memory



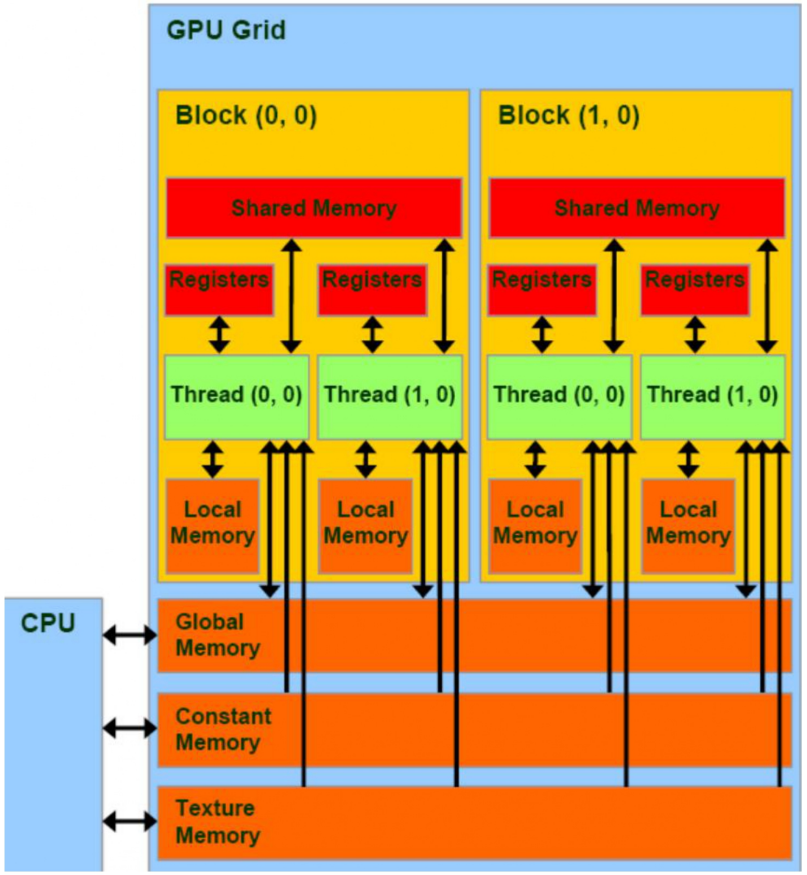
CUDA Memory Model Overview

- Global Memory
- Constant Memory



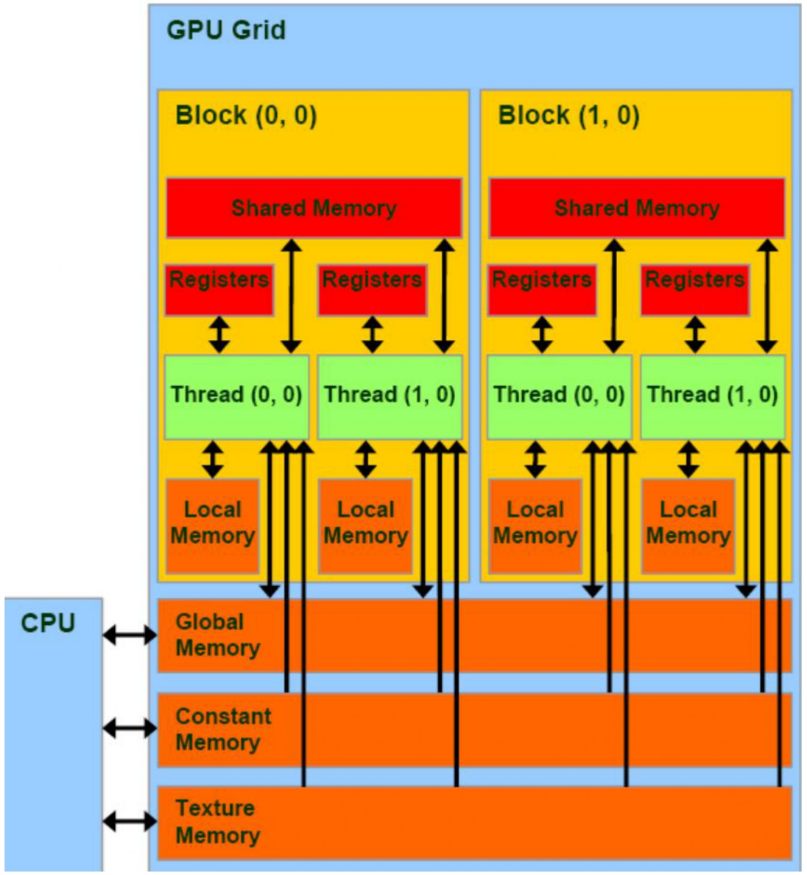
CUDA Memory Model Overview

- Global Memory
- Constant Memory
- Texture Memory



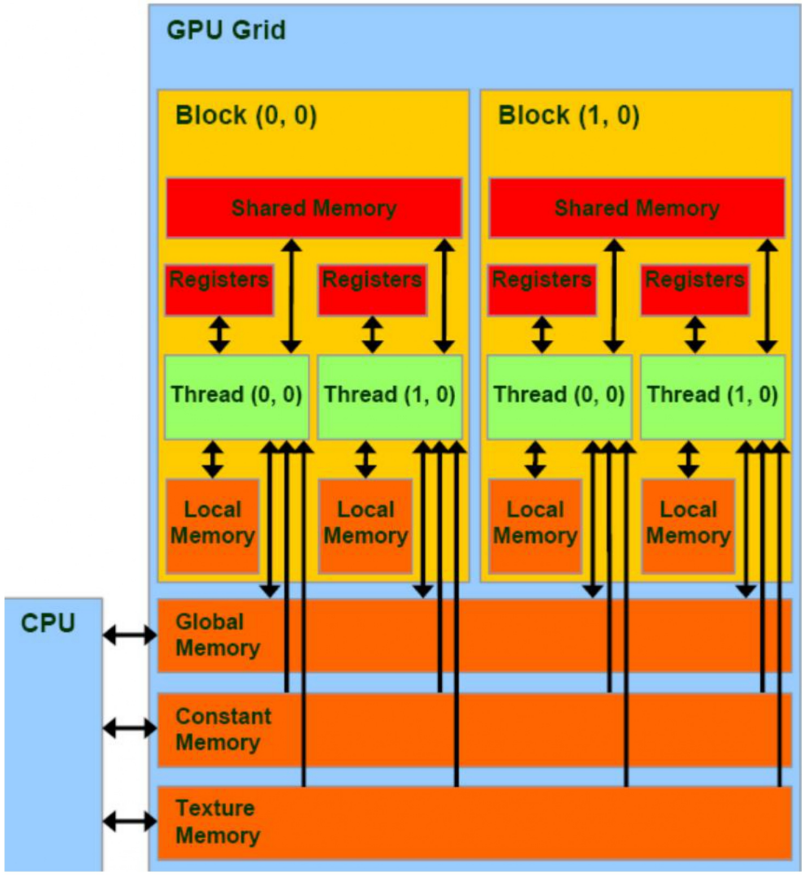
CUDA Memory Model Overview

- Global Memory
- Constant Memory
- Texture Memory
- L2 caches



CUDA Memory Model Overview

- Global Memory
- Constant Memory
- Texture Memory
- L2 caches
- Shared Memory / L1 cache



L1 cache vs shared Memory

- Host :

  cudaDeviceSetCacheConfig(kernelname, Parameter);

L1 cache vs shared Memory

- Host :

  cudaDeviceSetCacheConfig(kernelname, Parameter);

- Parameter :

  cudaFuncCachePreferNone

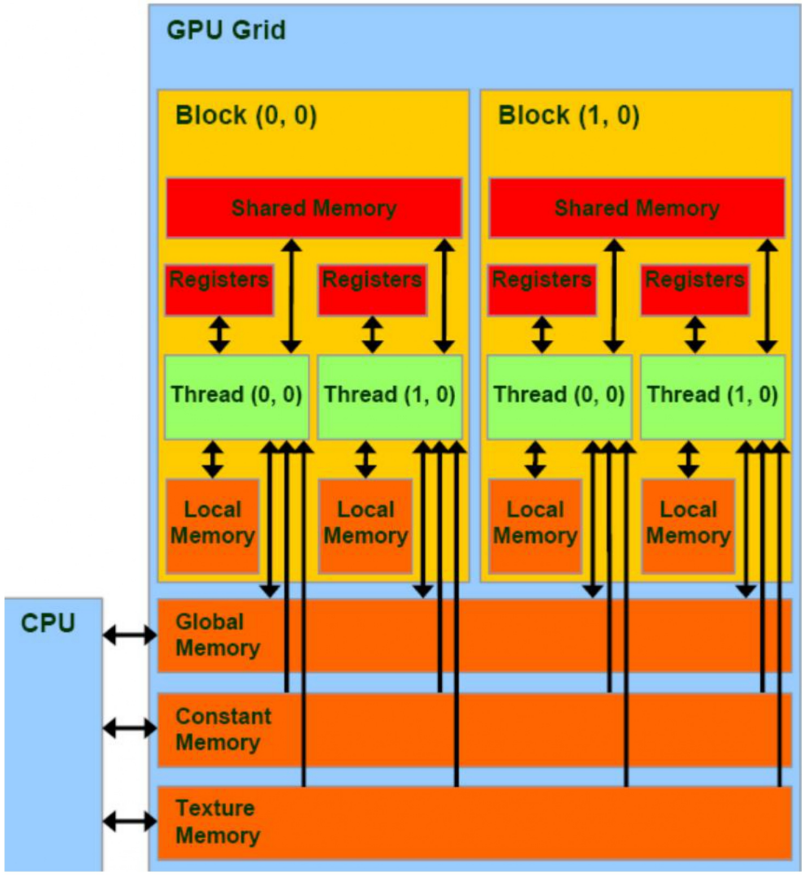
  cudaFuncCachePreferShared

  cudaFuncCachePreferL1

  cudaFuncCachePreferEqual

CUDA Memory Model Overview

- Global Memory
- Constant Memory
- Texture Memory
- L2 caches
- Shared Memory / L1 cache
- Registers



Memory Allocation Types

- There are three types of memory allocation in CUDA :
    1. Pageable memory
    2. Pinned memory
    3. Unified Memory

Pageable memory & Pinned memory

- The memory allocated in host is by default pageable memory

Pageable memory & Pinned memory

- The memory allocated in host is by default pageable memory
- CUDA allows to make host memory pinned

Pageable memory & Pinned memory

- The memory allocated in host is by default pageable memory
- CUDA allows to make host memory pinned
- CUDA allows direct access to pinned host memory from device

Pageable memory & Pinned memory

- The memory allocated in host is by default pageable memory
- CUDA allows to make host memory pinned
- CUDA allows direct access to pinned host memory from device
- Syntax :

```
cudaHostAlloc(&ptr,size,0);
```

Example-1

```
__global__ void Mykernel(int * count) {
printf("\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaHostAlloc(&count,sizeof(int),0);
Mykernel<<<1,1>>>(count);
CDS();
Printf("\ncounter from Host = %d ", ++*count);
return 0;
 }
```

Example-1

```
__global__ void Mykernel(int * count) {
printf("\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaHostAlloc(&count,sizeof(int),0);
Mykernel<<<1,1>>>(count);
CDS();
Printf("\ncounter from Host = %d ", ++*count);
return 0;
 }
```

Output :

counter from device = 1

counter from Host = 2

Example-2

```
__global__ void Mykernel(int * count) {
printf("\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaHostAlloc(&count,sizeof(int),0);
Mykernel<<<1,1>>>(count);
CDS();
Printf("\ncounter from Host = %d ", *count++);
return 0;
 }
```

Example-2

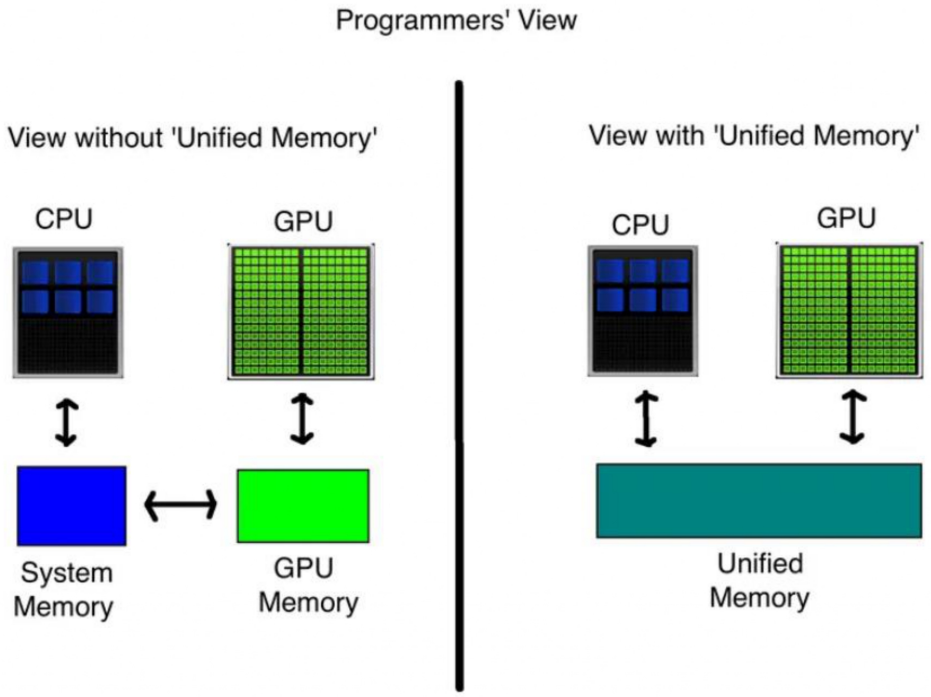
```
__global__ void Mykernel(int * count) {
printf("\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaHostAlloc(&count,sizeof(int),0);
Mykernel<<<1,1>>>(count);
CDS();
Printf("\ncounter from Host = %d ", *count++);
return 0;
 }
```

Output :

counter from device = 1

counter from Host = 1

Unified Memory



Unified Memory

- This creates a pool of managed memory where each allocation from this memory pool is accessible on both the host and the device with the same address or pointer.

Unified Memory

- This creates a pool of managed memory where each allocation from this memory pool is accessible on both the host and the device with the same address or pointer.

- Syntax :

__host__ cudaError_t **cudaMallocManaged** (void** devPtr, size_t size, unsigned int flags = cudaMemAttachGlobal)

Allocates memory that will be automatically managed by the Unified Memory system

Example-3

```
__global__ void Mykernel(int * count) {
printf(“\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaMallocManaged(&count,sizeof(int));
Mykernel<<<1,1>>>(count);
CDS();
Printf(“\ncounter from Host = %d “, *count++);
return 0;
 }
```

Example-3

```
__global__ void Mykernel(int * count) {
printf("\n counter from device = %d", ++*count);
}
int main() {
int *count =0;
cudaMallocManaged(&count,sizeof(int));
Mykernel<<<1,1>>>(count);
CDS();
Printf("\ncounter from Host = %d ", *count++);
return 0;
 }
```

Output :

counter from device = 1

counter from Host = 1

Thank You