CUDA Programming

Recap - AoS versus SoA

```
struct node {
        int a;
        double b;
        char c;
};
struct node allnodes[N];
```

```
struct node {
        int alla[N];
        double allb[N];
        char allc[N];
};
```

Expectation: When a thread accesses an attribute of a node, *it* also accesses *other attributes* of the *same node*.

Better locality (on CPU).

Expectation: When a thread accesses an attribute of a node, its *neighboring thread* accesses the *same attribute* of the *next node*.

Better coalescing (on GPU).

Shared Memory

- Programmable L1 cache / Scratchpad memory
- Accessible only in a thread block
- Useful for repeated small data or coordination

```
__shared__ float a[N];
__shared__ unsigned s;

a[id] = id;
if (id == 0) s = 1;
```

Classwork

- You are given a 1024x1024 integer matrix M.
- Each row is assigned to a thread block.
- Each thread is assigned a matrix element M[i][j].
- It changes M[i][j] to M[i][j] + M[i][j+1] (where possible).
- Exploit shared memory.

Shared Memory

```
#include <stdio.h>
#include <cuda.h>

#define BLOCKSIZE	1024

__global__ void dkernel() {
	__shared__ unsigned s;

	if (threadIdx.x == 0) s = 0;

	if (threadIdx.x == 1) s += 1;

	if (threadIdx.x == 100) s += 2;

	if (threadIdx.x == 0) printf("s=%d\n", s);
}
int main() {
	dkernel<<<1, BLOCKSIZE>>>();
	cudaDeviceSynchronize();
}
```

Shared Memory

```
#include <stdio.h>
#include <cuda.h>

#define BLOCKSIZE     1024

__global__ void dkernel() {
        __shared__ unsigned s;

        if (threadIdx.x == 0) s = 0;

        if (threadIdx.x == 1) s += 1;

        if (threadIdx.x == 100) s += 2;

        if (threadIdx.x == 0) printf("s=%d\n", s);
}
int main() {
        dkernel<<<1, BLOCKSIZE>>>();
        cudaDeviceSynchronize();
}
```

```
s=3
```

Shared Memory

```
#include <stdio.h>
#include <cuda.h>

#define BLOCKSIZE	1024

__global__ void dkernel() {
	__shared__ unsigned s;

	if (threadIdx.x == 0) s = 0;

	if (threadIdx.x == 1) s += 1;

	if (threadIdx.x == 100) s += 2;

	if (threadIdx.x == 0) printf("s=%d\n", s);
}
int main() {
	dkernel<<<2, BLOCKSIZE>>>();
	cudaDeviceSynchronize();
}
```

```
s=3
s=3
```

Shared Memory

```
#include <stdio.h>
#include <cuda.h>

#define BLOCKSIZE     1024

__global__ void dkernel() {
        __shared__ unsigned s;

        if (threadIdx.x == 0) s = 0;

        if (threadIdx.x == 1) s += 1;

        if (threadIdx.x == 100) s += 2;

        if (threadIdx.x == 0) printf("s=%d\n", s);
}
int main() {
        int i;
        for (i = 0; i < 10; ++i) {
                dkernel<<<2, BLOCKSIZE>>>();
                cudaDeviceSynchronize();
        }
}
```

```
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=3
s=1
s=3
s=3
s=3
```