Computational Science on Many-Core Architectures

360.252

Josef Weinbub
Karl Rupp

Institute for Microelectronics, TU Wien
http://www.iue.tuwien.ac.at/

Zoom Channel 621 2711 2607
Wednesday, November 8, 2023

Warp Shuffles

A Warp

- (typically) 32 threads in a CUDA thread block execute simultaneously
- they are called a *warp*
- no race conditions within a warp possible
- CUDA variable inside kernel: `warpSize` (compile time constant)

Warp Shuffles

A Warp

- (typically) 32 threads in a CUDA thread block execute simultaneously
- they are called a *warp*
- no race conditions within a warp possible
- CUDA variable inside kernel: `warpSize` (compile time constant)

Problem

- Exchanging data across threads via shared memory is relatively slow

Warp Shuffles

A Warp

- (typically) 32 threads in a CUDA thread block execute simultaneously
- they are called a *warp*
- no race conditions within a warp possible
- CUDA variable inside kernel: `warpSize` (compile time constant)

Problem

- Exchanging data across threads via shared memory is relatively slow

Solution

- Warp shuffle routines:
  - `__shfl_up_sync`
  - `__shfl_down_sync`
  - `__shfl_xor_sync`
  - `__shfl_sync`

Warp Shuffles

```
T __shfl_up_sync(unsigned mask, T var,unsigned int delta);
```

Move thread values to higher thread IDs

- `mask` controls which threads are involved — usually set to `-1` or `0xffffffff`, equivalent to all 1's
- `var` is a local register variable (int, unsigned int, long long, unsigned long long, float or double)
- `delta` is the offset within the warp — current thread value if offset runs out of bounds

Warp Shuffles

```
T __shfl_up_sync(unsigned mask, T var,unsigned int delta);
```

Move thread values to higher thread IDs

- `mask` controls which threads are involved — usually set to `-1` or `0xffffffff`, equivalent to all 1's
- `var` is a local register variable (int, unsigned int, long long, unsigned long long, float or double)
- `delta` is the offset within the warp — current thread value if offset runs out of bounds

```
T __shfl_down_sync(unsigned mask, T var,unsigned int delta);
```

Move thread values to lower thread IDs

Defined similarly

Warp Shuffles

```
T __shfl_xor_sync(unsigned mask, T var,unsigned int laneMask);
```

Move thread values to other XOR'd thread IDs

- an XOR (exclusive or) operation is performed between `laneMask` and the calling thread's `laneID` to determine the lane from which to copy the value
- (`laneMask` constrols the bits to be flipped within `laneID`)
- very useful for reductionoperations and FFTs

Warp Shuffles

```
T __shfl_xor_sync(unsigned mask, T var,unsigned int laneMask);
```

Move thread values to other XOR'd thread IDs

- an XOR (exclusive or) operation is performed between `laneMask` and the calling thread's `laneID` to determine the lane from which to copy the value
- (`laneMask` constrols the bits to be flipped within `laneID`)
- very useful for reductionoperations and FFTs

```
T __shfl_sync(unsigned mask, T var,unsigned int srcLane);
```

Get data from a different thread

copies data from the `srcLane` thread

Warp Shuffles

Warning

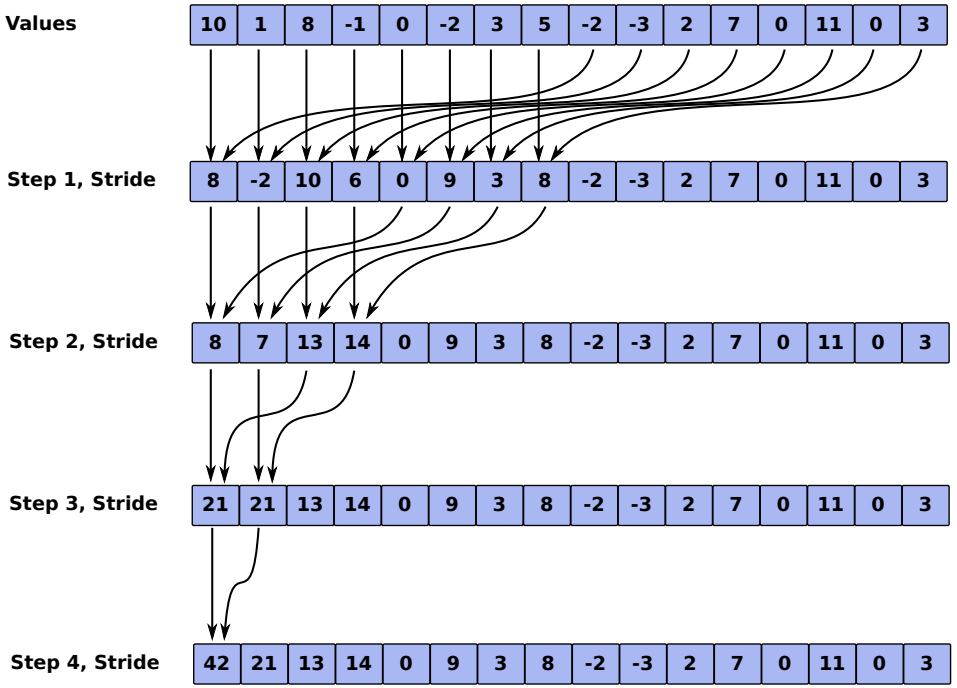
- Threads may only read data from another threadwhich is actively participating in the shuffle command.
- If the target thread is inactive, the retrieved value is undefined.
- Thus, be careful with conditional code!

Reference

- `https://people.maths.ox.ac.uk/gilesm/cuda/lecs/lec4.pdf`

Parallel Primitives

Reductions with Many Threads




```
__kernel my_warp_reduction(double *x) {
double value = x[threadIdx.x];
for (int i=16; i>0; i=i/2)
  value += __shfl_down_sync(-1, value, i);

// thread 0 contains sum of all values within the warp
}
```

Parallel Primitives

Another way to compute warp reductions

```
__kernel my_warp_reduction2(double *x) {
double value = x[threadIdx.x];
for (int i=16; i>0; i=i/2)
  value += __shfl_xor_sync(-1, value, i);

// all threads in the warp contain the warp sum
}
```