Technische Universität München
Lehrstuhl Informatik V
M. Bader, A. Pöppl

WS 2016/17
Solution - Worksheet 3
November 30th 2016

Tutorial: HPC - Algorithms and Applications WS 16/17

Worksheet 3 Solution: Coalesced Access

T3.1: Recap on Coalesced Access

Consider this CUDA kernel call:

```
dim3 grid(8, 8, 1); dim3 block(32, 32, 1);
kernel<<<grid, block>>>(A);

__global__ void kernel(float* A) {
    int tx = threadIdx.x, ty = threadIdx.y, tz = threadIdx.z;
    const int n = 64;
    float f;
    /* set value of f here */
}
```

For each of the following instructions, answer shortly if access to the array `A` is uncoalesced, partially coalesced or coalesced (chipset: NVidia Fermi, CUDA cc ≥ 2.0).

a) `f = A[tx];` **Coalesced**. a single warp will only access data from one 128 byte segment.

b) `f = A[(tx * n + ty) * n + tz];` **Uncoalesced**. The memory is being accessed out of sequence, addresses accessed by a single warp will be spread out over $>$ 128kB.

c) `f = A[tx + 1];` **Partially coalesced**. For the first warp, two accesses are generated. Second warp may use cached data from previous load, and must perform only one load.

d) `f = A[ty];` **Coalesced**. Each warp only has to load a single 128 byte segment. However. The bandwidth of the memory isn't used to its fullest, as only four byte out of the 128 bytes loaded will be used.

e) `f = A[2 * tx];` **Partially coalesced**. Each warp has to load two segments.

f) `f = A[n * tx];` **Uncoalesced**. For each thread a segment has to be loaded. Subsequently executed warps may use cached values.

g) `f = A[tx / 2 + 16];` **Coalesced**. Each warp has to load a single 128 byte segment.

h) `f = A[ty * tx];` **Partially Coalesced - Uncoalesced**. For each warp, `ty` segments have to be loaded.