master hpc4ai_study / demo_cuda / bitonic_sort.cu

Go to file t

Notflowing add bitonic sort and radix select topk a1555f1 · 3 minutes ago History

Code Blame 55 lines (50 loc) · 2.2 KB Raw

```
1   // bitonic sort
2   ////////////////////////////////////////////////////////////////////////////////
3   // Monolithic bitonic sort kernel for short arrays fitting into shared memory
4   ////////////////////////////////////////////////////////////////////////////////
5   __global__ void bitonicSortShared(uint *d_DstKey, uint *d_DstVal,
6                                     uint *d_SrcKey, uint *d_SrcVal,
7                                     uint arrayLength, uint dir) {
8     // Handle to thread block group
9     cg::thread_block cta = cg::this_thread_block();
10    // Shared memory storage for one or more short vectors
11    __shared__ uint s_key[SHARED_SIZE_LIMIT];
12    __shared__ uint s_val[SHARED_SIZE_LIMIT];
13
14    // Offset to the beginning of subbatch and load data
15    d_SrcKey += blockIdx.x * SHARED_SIZE_LIMIT + threadIdx.x;
16    d_SrcVal += blockIdx.x * SHARED_SIZE_LIMIT + threadIdx.x;
17    d_DstKey += blockIdx.x * SHARED_SIZE_LIMIT + threadIdx.x;
18    d_DstVal += blockIdx.x * SHARED_SIZE_LIMIT + threadIdx.x;
19    s_key[threadIdx.x + 0] = d_SrcKey[0];
20    s_val[threadIdx.x + 0] = d_SrcVal[0];
21    s_key[threadIdx.x + (SHARED_SIZE_LIMIT / 2)] =
22        d_SrcKey[(SHARED_SIZE_LIMIT / 2)];
23    s_val[threadIdx.x + (SHARED_SIZE_LIMIT / 2)] =
24        d_SrcVal[(SHARED_SIZE_LIMIT / 2)];
25
26    for (uint size = 2; size < arrayLength; size <<= 1) {
27      // Bitonic merge
28      uint ddd = dir ^ ((threadIdx.x & (size / 2)) != 0);
29
30      for (uint stride = size / 2; stride > 0; stride >>= 1) {
31        cg::sync(cta);
32        uint pos = 2 * threadIdx.x - (threadIdx.x & (stride - 1));
33        Comparator(s_key[pos + 0], s_val[pos + 0], s_key[pos + stride],
34                   s_val[pos + stride], ddd);
35      }
36    }
37
38    // ddd == dir for the last bitonic merge step
39    {
40      for (uint stride = arrayLength / 2; stride > 0; stride >>= 1) {
41        cg::sync(cta);
42        uint pos = 2 * threadIdx.x - (threadIdx.x & (stride - 1));
43        Comparator(s_key[pos + 0], s_val[pos + 0], s_key[pos + stride],
44                   s_val[pos + stride], dir);
45      }
46    }
47
48    cg::sync(cta);
49    d_DstKey[0] = s_key[threadIdx.x + 0];
50    d_DstVal[0] = s_val[threadIdx.x + 0];
51    d_DstKey[(SHARED_SIZE_LIMIT / 2)] =
52        s_key[threadIdx.x + (SHARED_SIZE_LIMIT / 2)];
53    d_DstVal[(SHARED_SIZE_LIMIT / 2)] =
54        s_val[threadIdx.x + (SHARED_SIZE_LIMIT / 2)];
55  }
```