Quickfind (max top_k)

Brief

Use quicksort method to select top k maximum elements

Method

The method is similar quick sort. But instead we sort the array, we only need to find the pivot that can split the array into two parts. One is all bigger than pivot, the other is all smaller than pivot and have size `k`

1. find a pivot and split the array into two parts.
2. find out which part the k is.
3. sort the part contains the k elements.
4. repeat step1,2,3 until one part contains exactly k elements.

Since the per thread operates one element, the algorithm is pretty fast.

Usage

```
qsort_warp(unsigned *indata,
    unsigned *outdata,
    unsigned int offset,
    unsigned int len,
    qsortAtomicData *atomicData,
    qsortRingbuf *atomicDataStack,
    unsigned int source_is_indata,
    unsigned int depth,
    int k,
    unsigned *target_gpu)
```

- **unsigned** `*indata`: the data need to be sorted
- **unsigned** `*outdata`: the data is already sorted
- **unsigned int** `offset`: the beginning of the array
- **unsigned int** `len`: the length of parts to be sorted
- **qsortAtomicData** `*atomicData`: the offset for each element in a warp
- **qsortRingbuf** `*atomicDataStack`: A ring-buffer for rapid stack allocation
- **unsigned int** `depth`: the recursive layer of quickfind
- **int** `k`: the elements need to be find
- **unsigned** `*target_gpu`: the kth element

Test

```
[hadoop@yj152 yjb]$ ./a.out
Running qsort on 200000000 elements
cpu target:199999951
time for cpu quicksort is 785.523 ms
gpu target:199999951
time for gpu quicksort is 70.828 ms (2823.741 Melems/sec)
[hadoop@yj152 yjb]$ ./a.out
Running qsort on 200000000 elements
cpu target:199999951
time for cpu quicksort is 784.586 ms
gpu target:199999951
time for gpu quicksort is 71.019 ms (2816.159 Melems/sec)
[hadoop@yj152 yjb]$ ./a.out
Running qsort on 200000000 elements
cpu target:199999951
time for cpu quicksort is 785.932 ms
gpu target:199999951
time for gpu quicksort is 70.640 ms (2831.271 Melems/sec)
```

In the test our quickfind in gpu can find top_k in 200000000 elements during 70-90ms, **10X** faster than cpuquickfind!