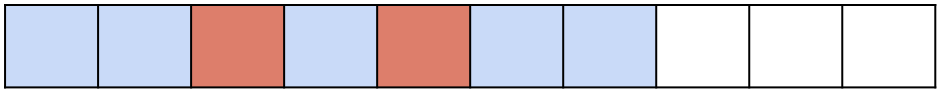
Tech Talk 2:
Dumb multithreading algorithms

Henry Huang

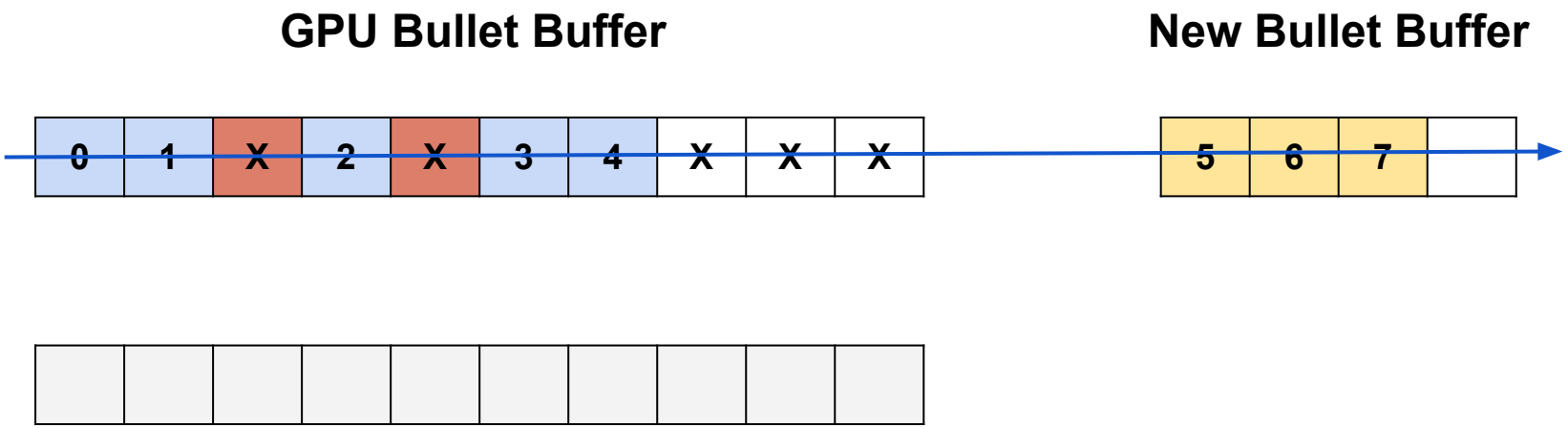
E190U

Last Episode…

GPU Bullet Buffer



New Bullet Buffer

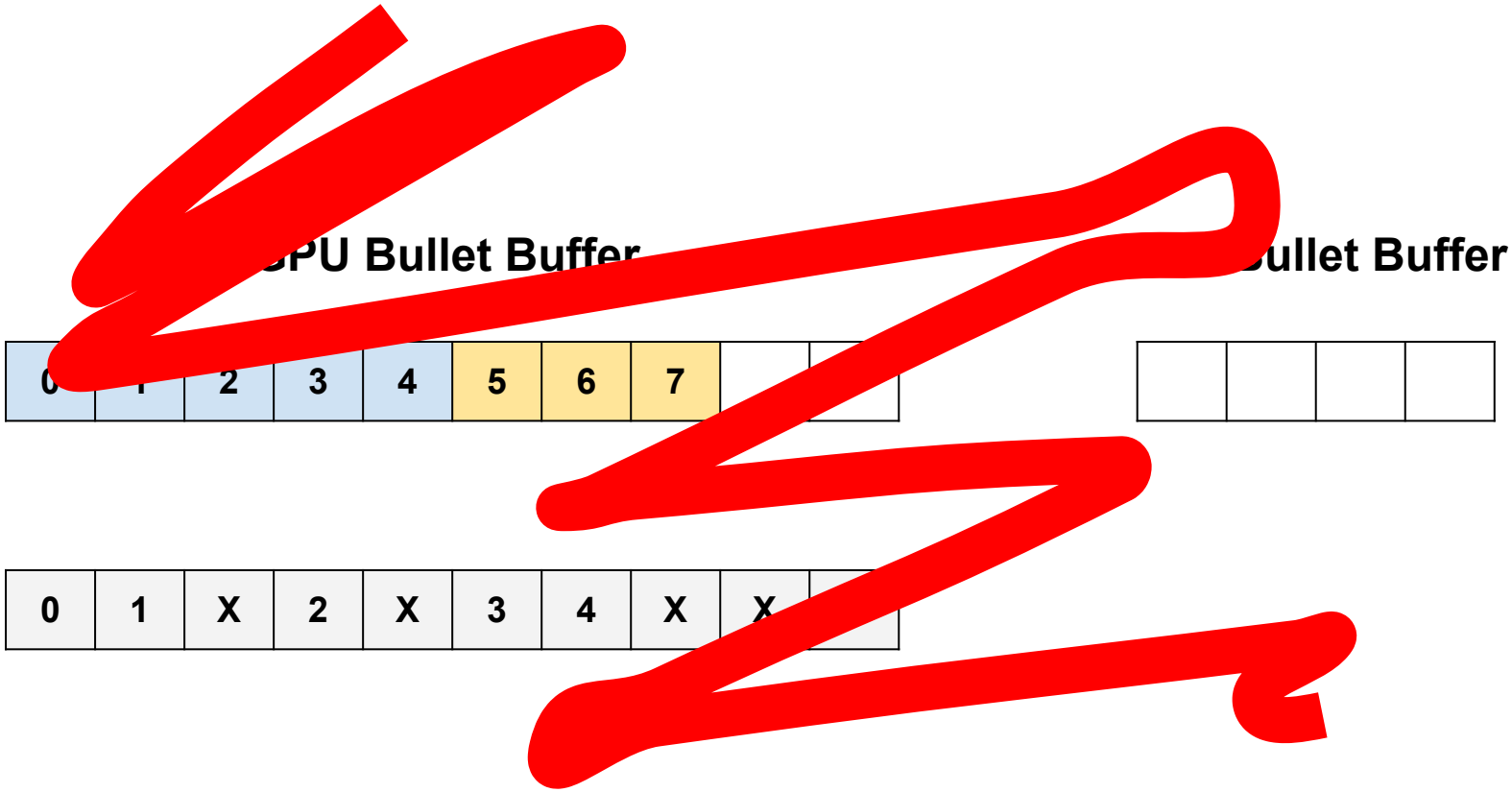

GPU Bullet Buffer
New Bullet Buffer
0
1
X
2
X
3
4
X
X
X
5
6
7

GPU Bullet Buffer

0	1	2	3	4	5	6	7		

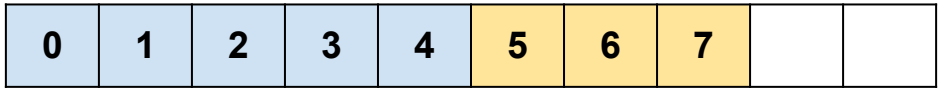
0	1	X	2	X	3	4	X	X	X

New Bullet Buffer

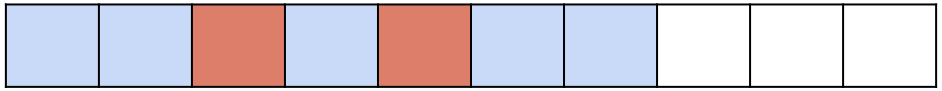

PU Bullet Buffer
ullet Buffer
0
1
2
3
4
5
6
7
0
1
X
2
X
3
4
X
X

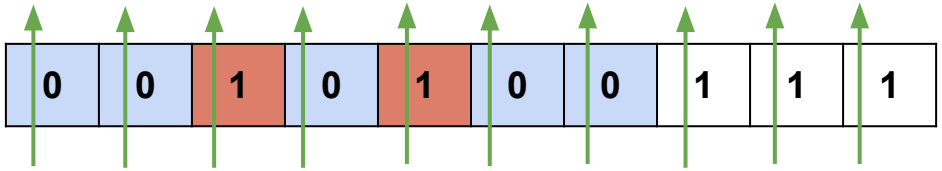
- No more single-threading bottleneck (on GPU)
- No more duplicate memory

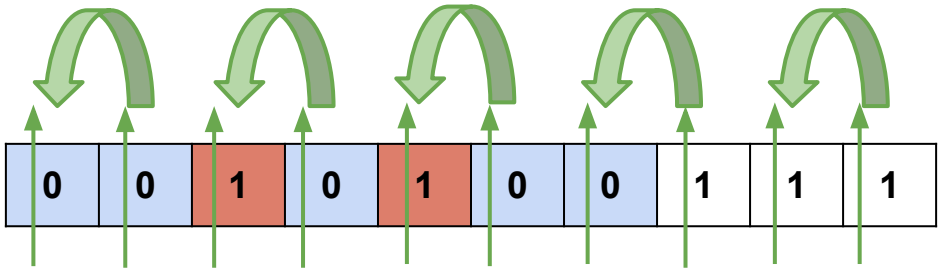
GPU Bullet Buffer

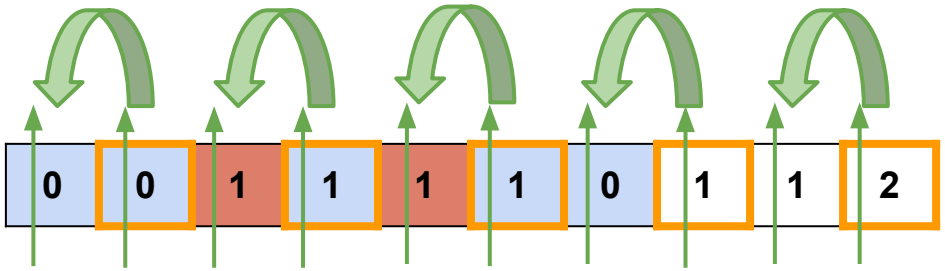


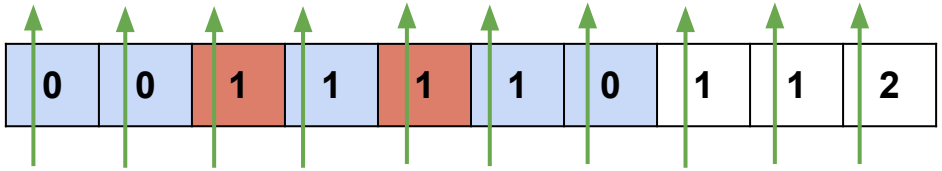
GPU Bullet Buffer

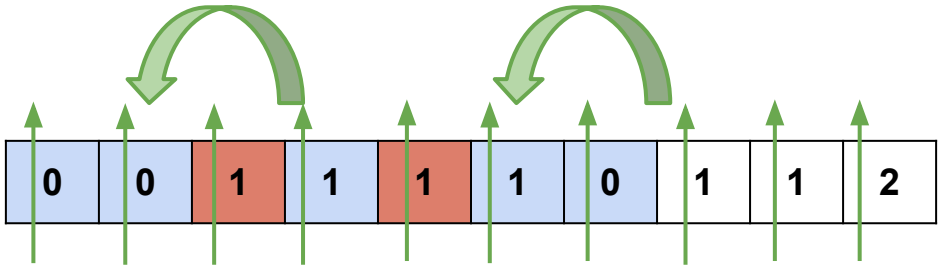


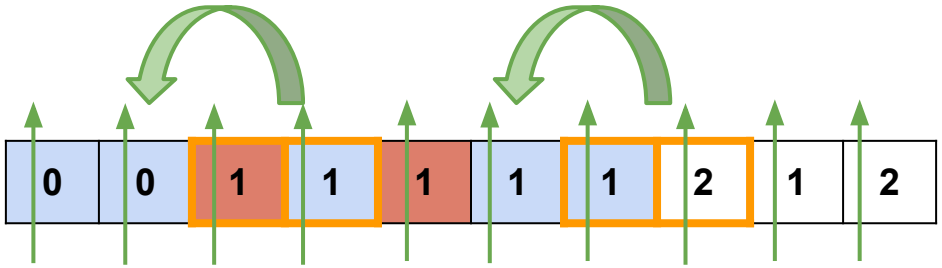

0
0
1
0
1
0
0
1
1
1

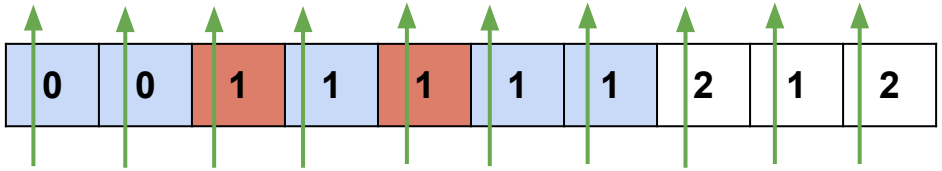

0
0
1
0
1
0
0
1
1
1

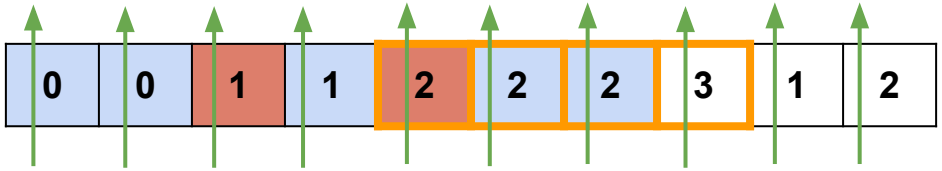

0
0
1
1
1
1
0
1
1
2

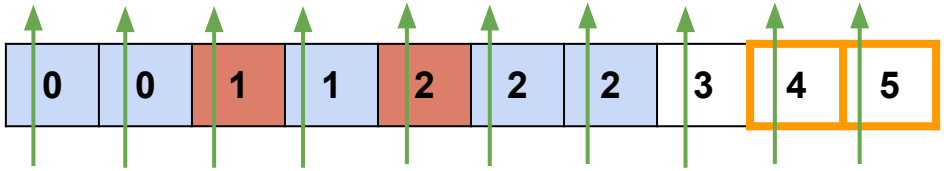

0
0
1
1
1
1
0
1
1
2


0
0
1
1
1
1
0
1
1
2

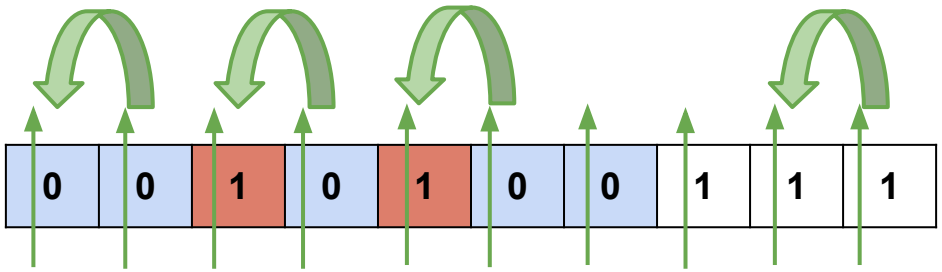

0
0
1
1
1
1
1
2
1
2

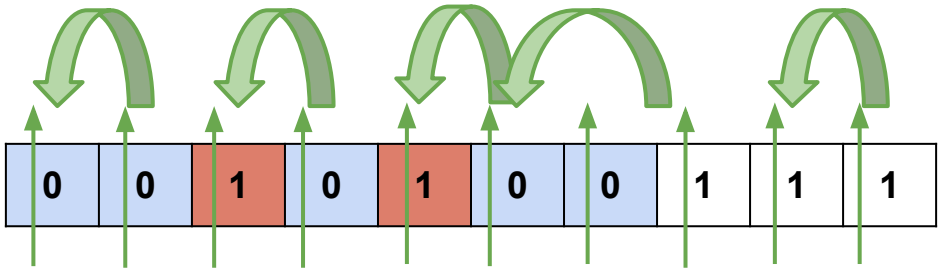

0
0
1
1
1
1
1
2
1
2

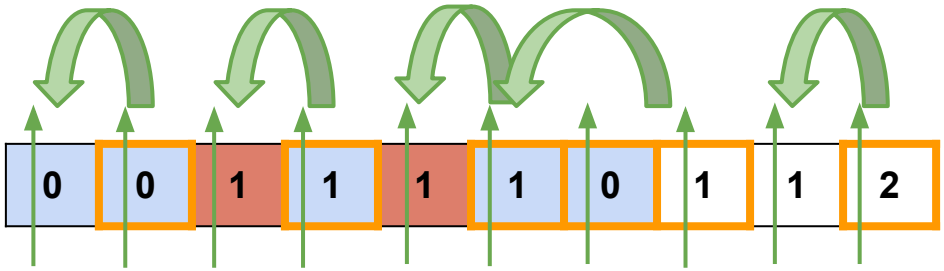

0
0
1
1
2
2
2
3
1
2


0
0
1
1
2
2
2
3
4
5

- Original Algorithm:
  - O(n)
- Multithread Algorithm:
  - O(log(n))


0
0
1
0
1
0
0
1
1
1


0
0
1
0
1
0
0
1
1
1


0
0
1
1
1
1
0
1
1
2

__syncthreads()

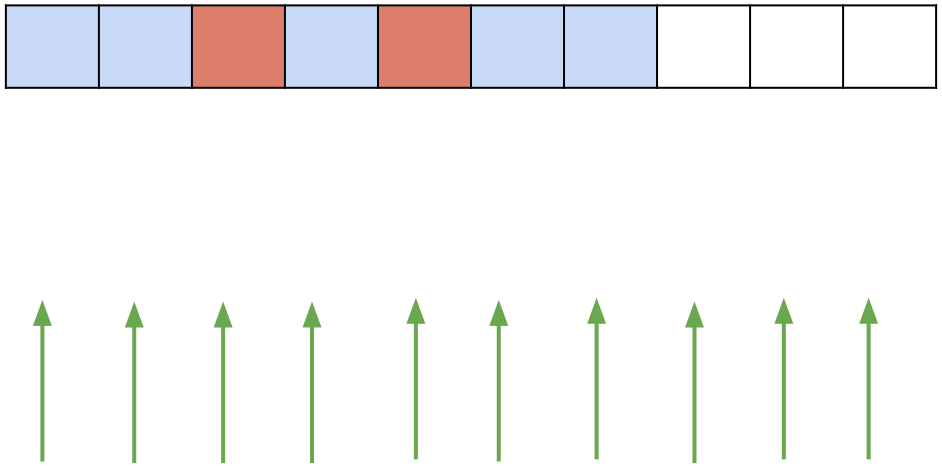
- Usage:

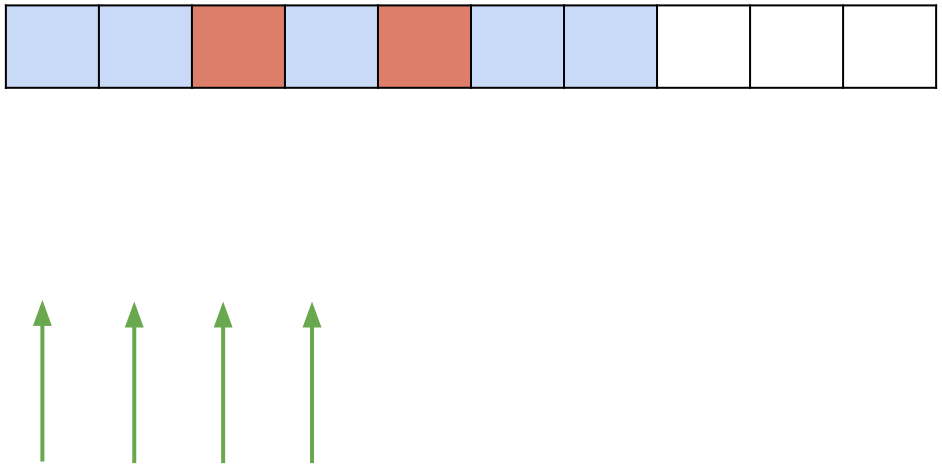
```
__global__ foo(...){
loop1(...);
__syncthreads();
loop2(...);
}
```

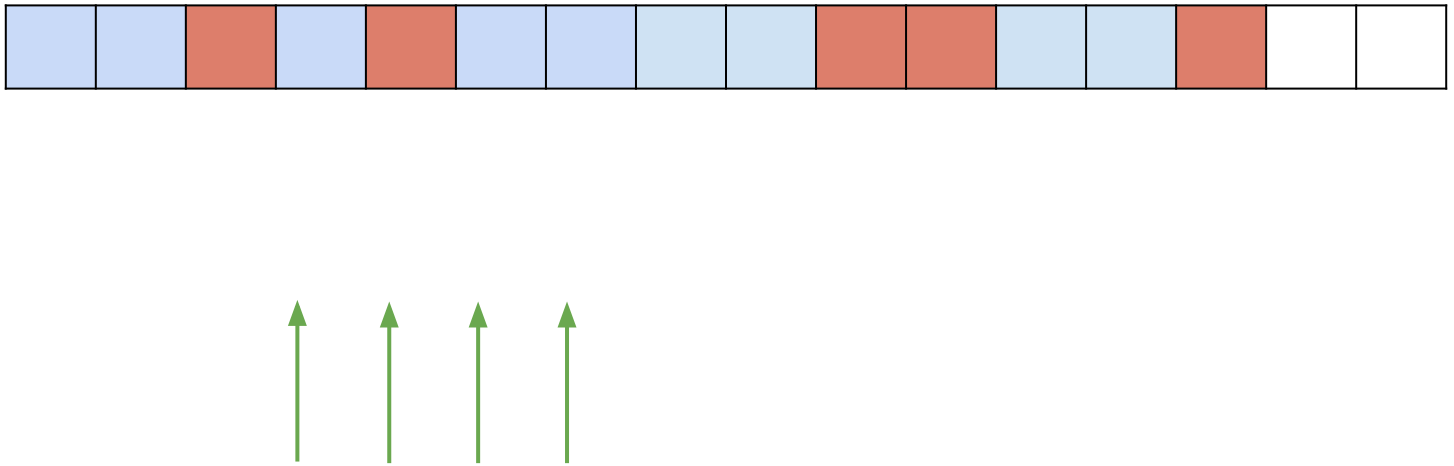
__syncthreads()

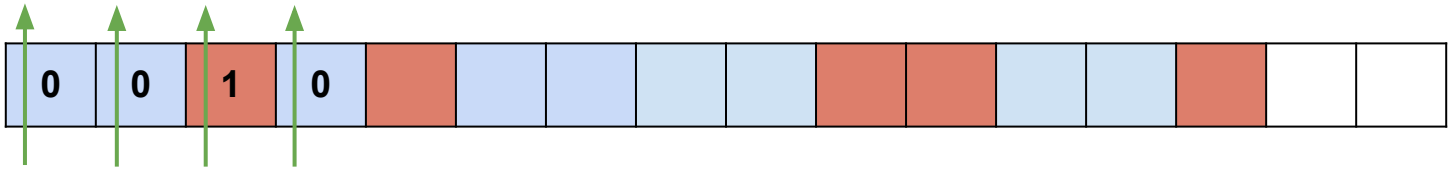
- Ensures all threads in a block will stop
- No effect outside of block
- Can cause deadlock if misused:

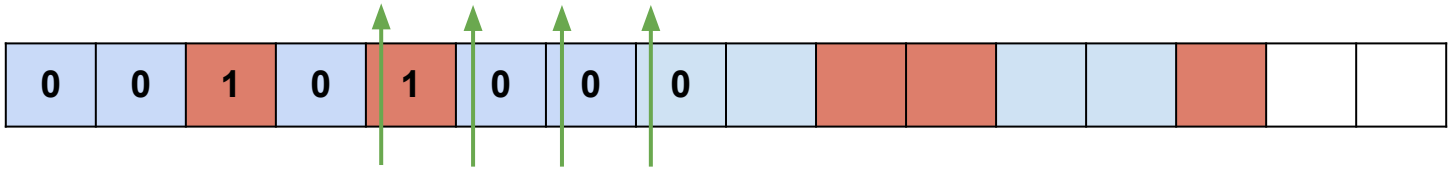
```
if(<non-trivial conditional>){
    __syncthreads(); //requires ALL threads to go
}
```



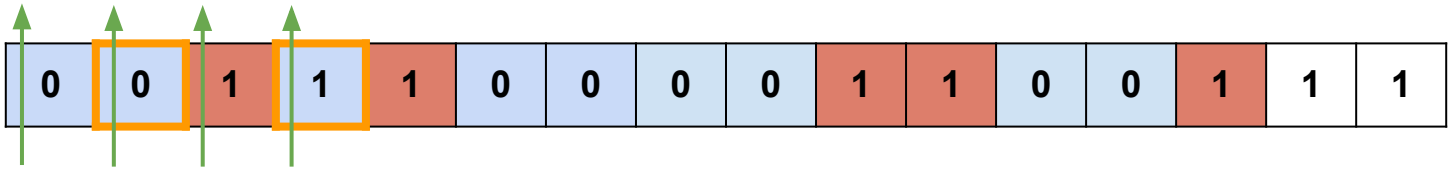


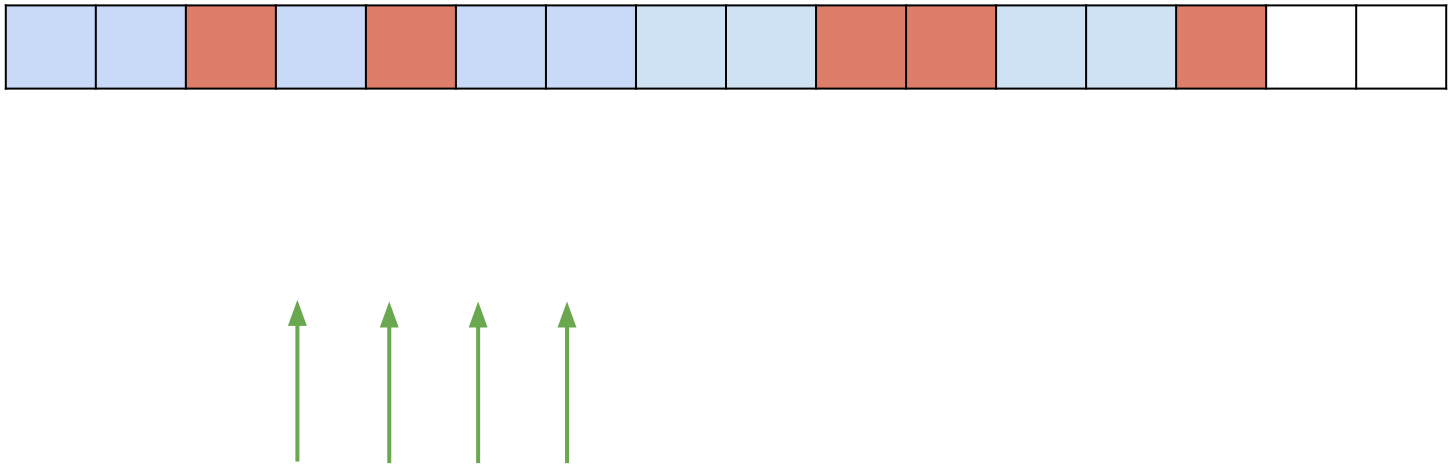


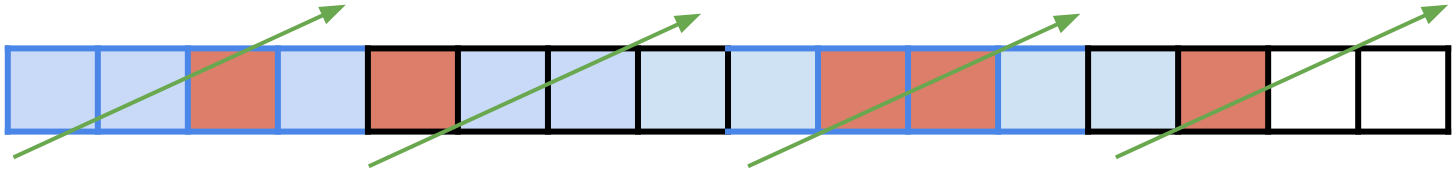

0
0
1
0

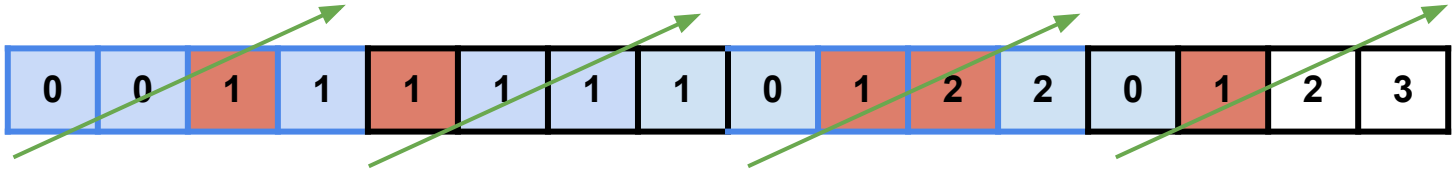

0
0
1
0
1
0
0
0

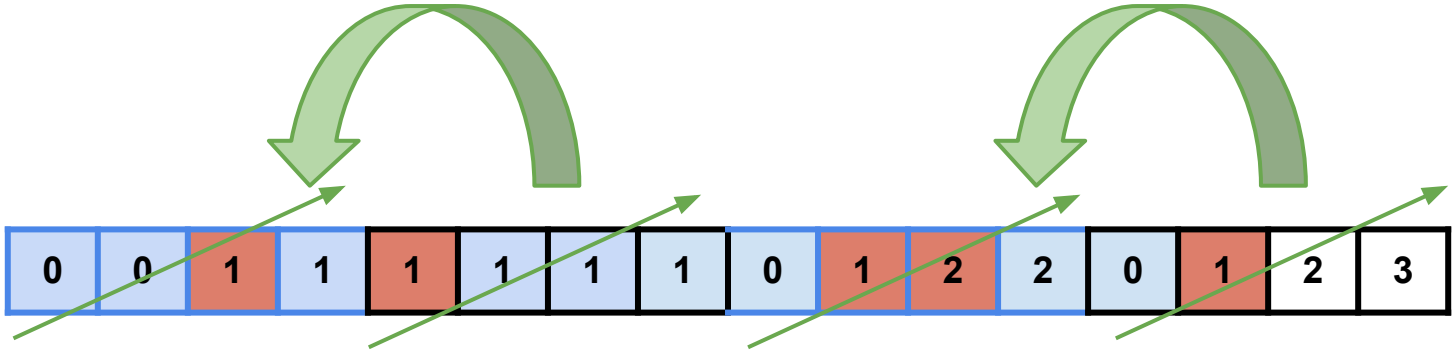
O(N/K*log(N))

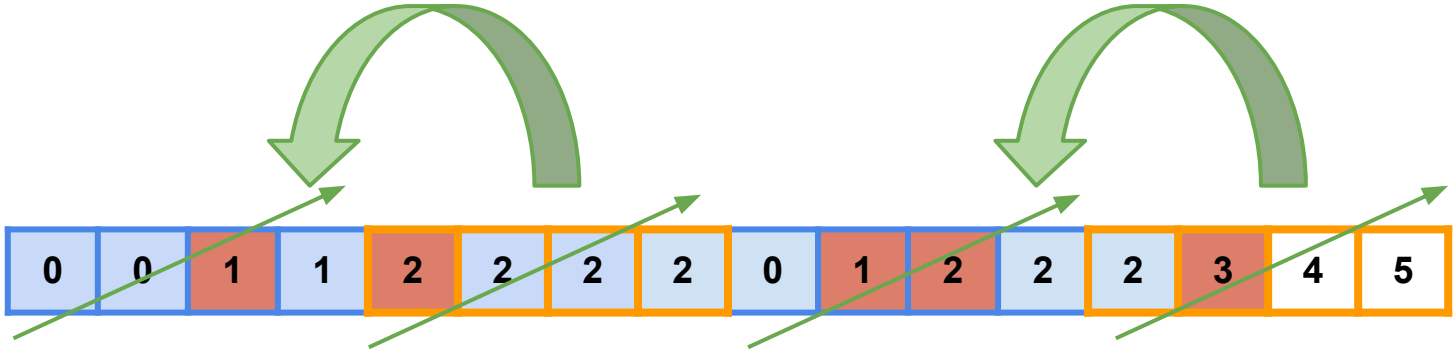




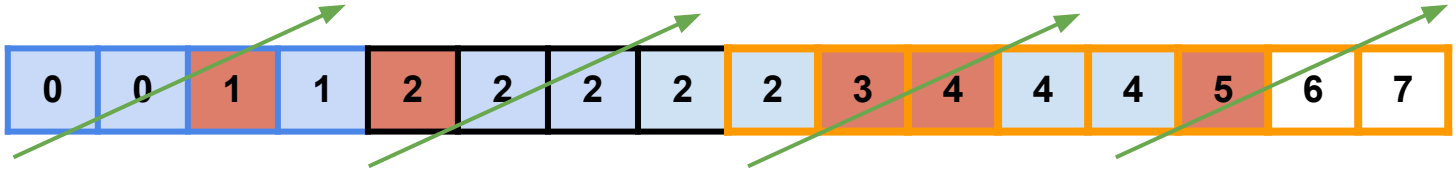



0
0
1
1
1
1
1
1
0
1
2
2
0
1
2
3

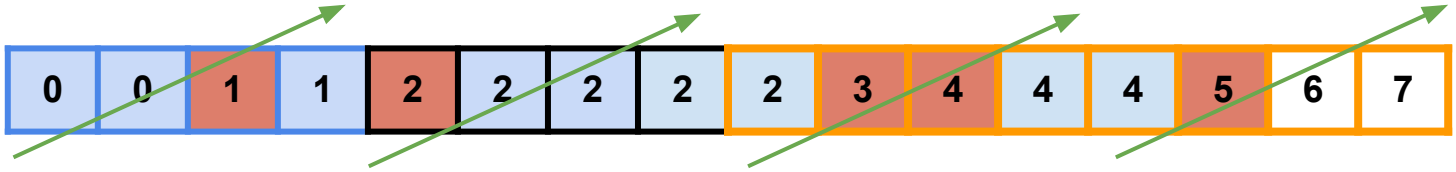

0
0
1
1
1
1
1
1
0
1
2
2
0
1
2
3


0
0
1
1
2
2
2
2
0
1
2
2
2
3
4
5

O(N/K*log(K))

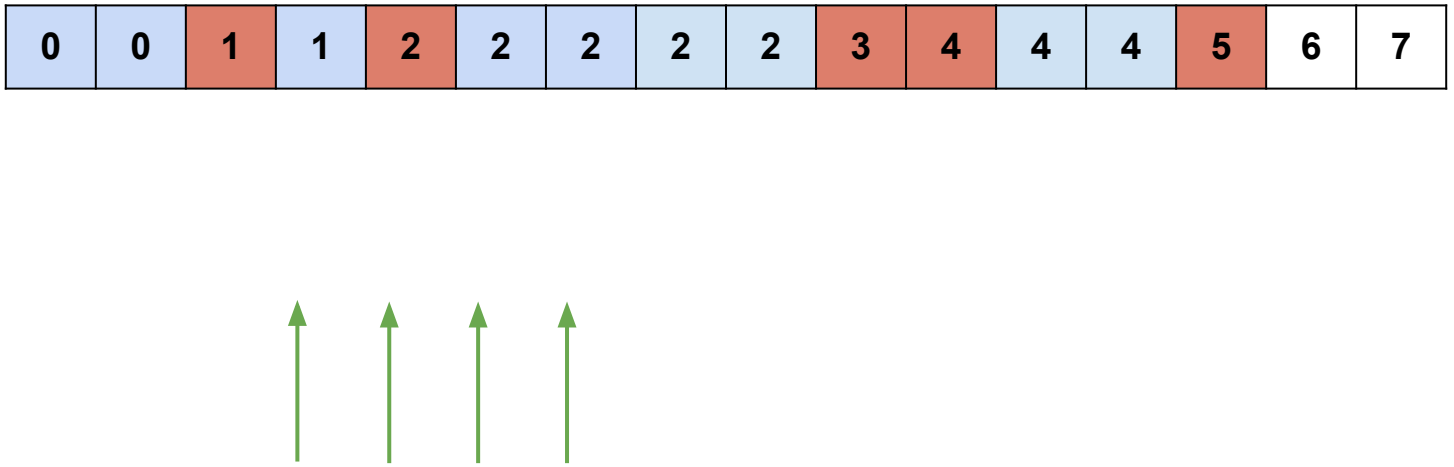


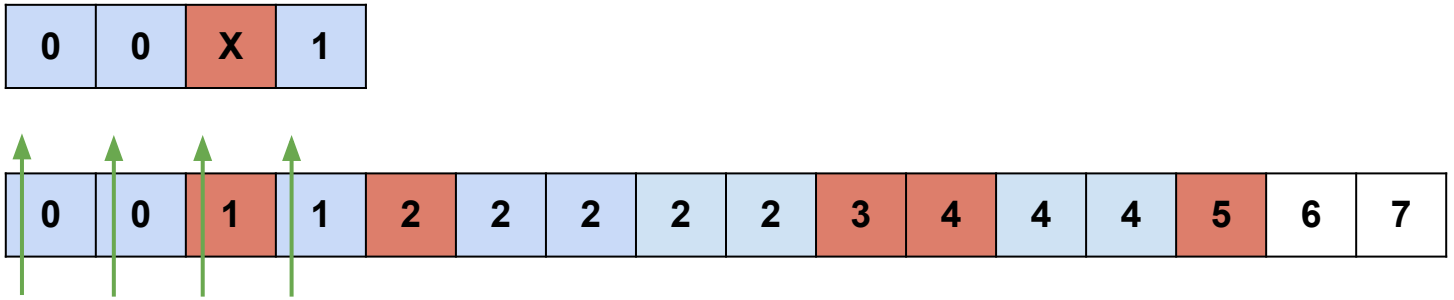
O((log(K)/K)*N)

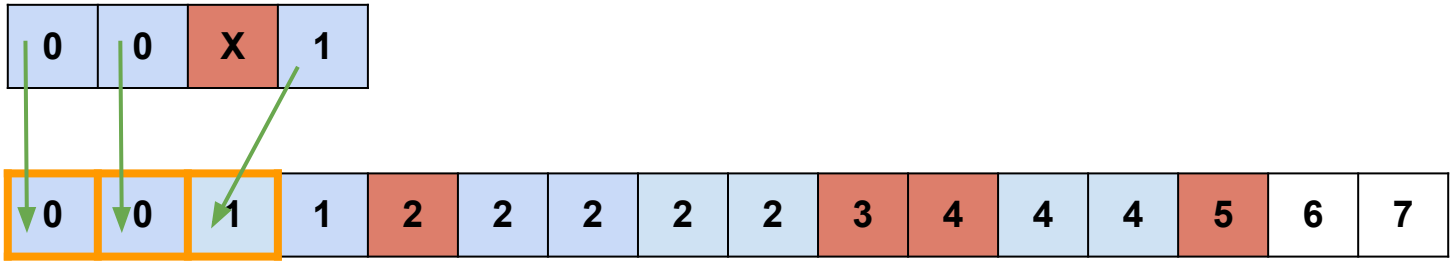


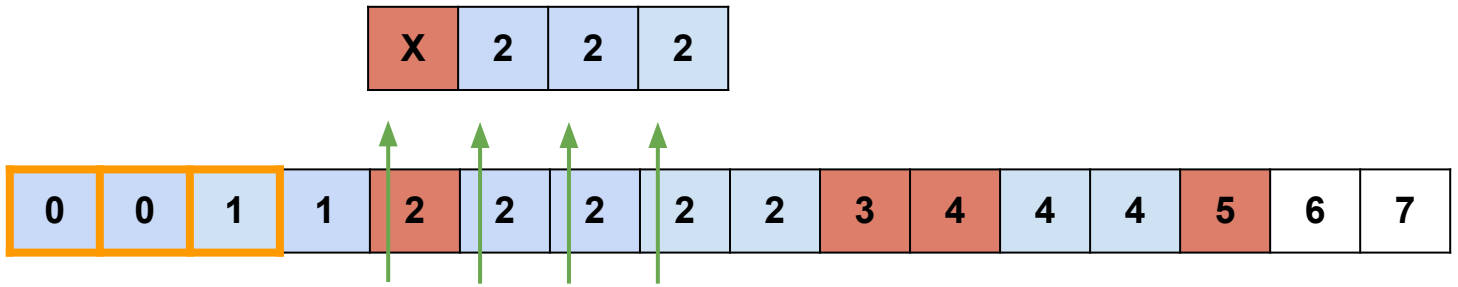
__syncthreads()

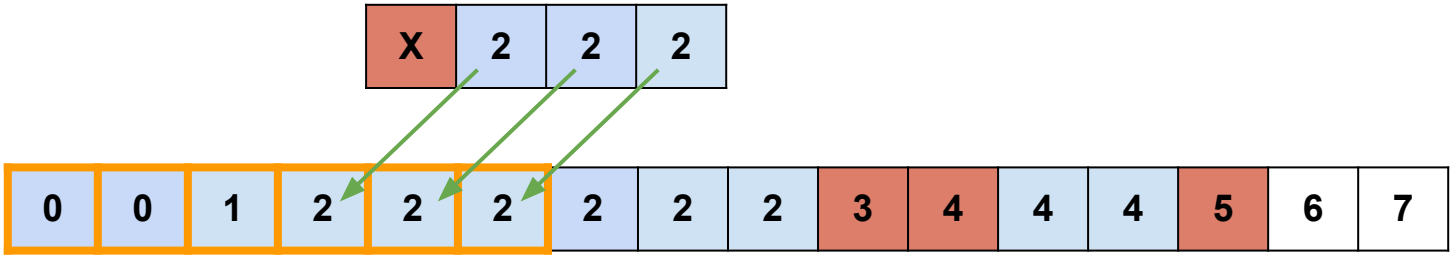
- Reorganizing an array in parallel safely


0
0
1
1
2
2
2
2
2
3
4
4
4
5
6
7


0
0
X
1
0
0
1
1
2
2
2
2
2
3
4
4
4
5
6
7


0
0
X
1
0
0
1
1
2
2
2
2
2
3
4
4
4
5
6
7


X
2
2
2
0
0
1
1
2
2
2
2
2
3
4
4
4
5
6
7


X
2
2
2
0
0
1
2
2
2
2
2
2
3
4
4
4
5
6
7

0
0
1
2
2
2
2
4
4
4
4
4
4
5
6
7