key points

histogtam for calculate the accumulated number of data.

atomic aperation: every thread holds on a single data. Making a series of operation be an undevided opeation by locking other operations with the function: int atomicAdd(int* adress, int val); replace the function like: *adress++;

For speeding up the accessing memory speed, placing data in the last-level cache(shared in SMs) can help decrease latency form hunhreds of cycles to tens of cycles.(differ from global memory)

Privatization: avoid traffic on a single target, cghange the aim target to shared memory, thus increase accessing speed, moreover, don't forget the function __syncthreads();

Coarsening: too many copies for each block submit form private to public may takes lots of time, so coarsening can help decreasing number of pribate copies. Interleaved partitioning and non-interleaved partitioning: for the previous one, the improved cache utilization and reduced memory latency due to better coalescing.

Aggregation: Consecutive similar data enhances memory access patterns and reduces contention, leading to more effective aggregation. Conversely, randomly sorted data can hinder performance due to poor memory coalescing and increased contention on shared resources.

1

1.1

~~1~~ for expressing maximum throughput, represent it in operations per second, which is 10 million operations per second.

1.2

~~for about 0.9*1/4ns+0.1*1/100ns * 1e9 per second~~ in real process, L2 and global aren't executed pipelined, they actually serialized, it should be thought together: $\frac{1}{(0.9*4ns)+(0.1*100ns)} = \frac{1}{3.6+10ns} \approx 7.35 \times 10^7$ operations/second.

1.3

it is 5*10 million bytes(50MFLOPs/s), the amount of throughput is the number of floating-point operations per second.

1.4

it should be about $\frac{1}{1ns}/1.1 * 5$

1.5

d

the use of this function is: atomicAdd(address, value);

1.6

1.6.1

~~524288/1024 times.~~ Total atmiuc ioerations is just 524288, instead of the number block.

1.6.2

52488/(52488/1024) times.

1.6.3

52488/[(52488/1024)*4] times.