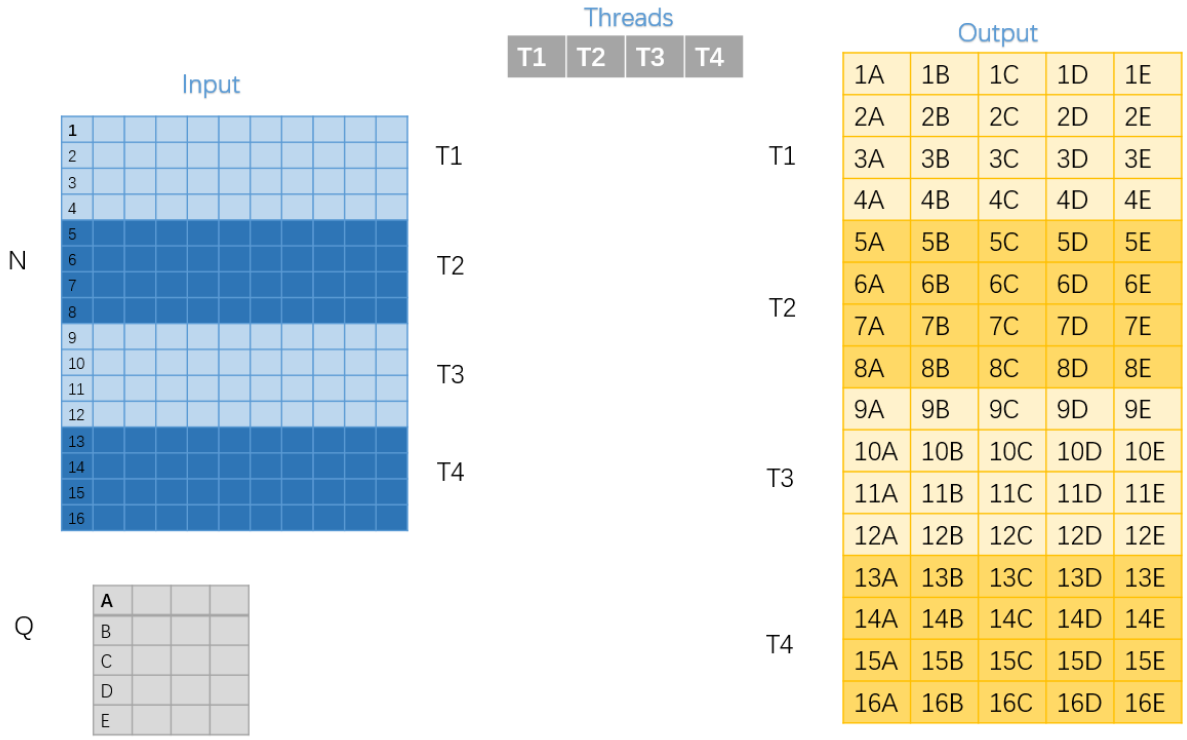
How the Parallel Computing is realized in FastLSH

The FastLSH uses its parallel computing power to accelerate LSH process.

This file introduces how the parallel computing is realized in FastLSH. Usually the multithread computing won't be very close to the ideal speed, because of the overhead of it. However, the FastLSH realizes a very ideal speedup. The principle is to make **each task totally independent** so the multithread program won't **be bothered by race condition**. There is no overhead from Lock.

The graph below is an example of the parallel computing in FastLSH, most of the parallel computing operation in FastLSH uses the same methods.



In this example, we have four threads. The input contains one big table N, and one small table Q, the output is a NxQ table whose values are computed from table N, Q. (This is the common scenario of the compute in FastLSH)

We use three methods to assign tasks to each thread.

Method 1: Give every thread access to the small table.

Method 2: Divided the Big table into segments, each thread has access to only one segment.

Method 3: Each thread only has access to a portion of the output table accords to the segment assigned in method 2.

In the graph above, the table N is divided into four segments T1,T2, T3 and T4. The four threads access one of the segment. Also all four threads have access to the table Q. In the output table, Thread 1 will only write to the T1 portion, Thread 2 will only write to the T2 portion and so on. Each thread won't access same part of the table at the same time, so each task is totally independent. We do not need lock to avoid race condition, the overhead of multithread is almost zero. So the parallel computing power is almost in full speed.