```
#include <libnvcd.h> // Required header inclusion

template <typename T>
__global__ void kernelX(T const * __restrict__ const a,
T * __restrict__ const b, int len)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    if (idx < len)
        b[idx] = a[idx];
}
template <typename T>
__global__ void kernelY(T const * __restrict__ a,
T const * __restrict__ b,
T * __restrict__ const c,
T scalar, int len)
{
    int idx = threadIdx.x + blockIdx.x * blockDim.x;
    if (idx < len)
        c[idx] = a[idx] + scalar * b[idx];
}

int host_function(int blockSize, in N){
    real *d_a, *d_b, *d_c;

    libnvcd_load(); //Init function for lbnvcd

    /* Compute execution configuration */
    dim3 dimBlock(blockSize);
    dim3 dimGrid(N/dimBlock.x );
    // ... Other computation
    /* Initialize memory on the device */
    set_array<real><<<dimGrid,dimBlock>>>(d_a, 2.f, N);
    set_array<real><<<dimGrid,dimBlock>>>(d_b, .5f, N);
    set_array<real><<<dimGrid,dimBlock>>>(d_c, .5f, N);
    // ... Other computation

    libnvcd_begin("kernelX"); //Begin couting, provide any name to denote the region
    kernelX<real><<<dimGrid,dimBlock>>>(d_a, d_c, N);
    libnvcd_end(); // End counting

    libnvcd_begin("kernelY"); //Begin couting, provide any name to denote the region
    kernelY<real><<<dimGrid,dimBlock>>>(d_b, d_c, d_a, scalar,  N);
    libnvcd_end(); // End counting

}
```