```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <sys/time.h>
#include "time.h"

using namespace std;

__global__ void parMap(float *pD, float *netD, int grid)
{
        unsigned int rID= blockDim.x*blockIdx.x + threadIdx.x;
        int left, right, top, bottom;
        float x,y, fL,fR,fB,fT;

        x = pD[rID*2];
        y = pD[rID*2+1];

        left = (int)floorf(x);
        right = left + 1;
        bottom = (int)floorf(y);
        top = bottom +1;

        if (left>= grid||right>= grid||top>= grid||bottom>= grid){
            left=0;
            right=1;
            top=1;
            bottom = 0;
            x=0.500000;
            y=0.500000;
        }

        fL = x - left;
        fR = 1 - fL;
        fB = y - bottom;
        fT = 1 - fB;

        netD[grid*left + bottom]  = netD[grid*left + bottom] +(fT*fR);
        netD[grid*right + bottom] = netD[grid*right + bottom]+(fT*fL);
        netD[grid*left+ top]      = netD[grid*left + top]    +(fB*fR);
        netD[grid*right+ top]     = netD[grid*right + top]   +(fB*fL);
}

int main(int argc, char *argv[])
{
    //--------------------Declaring Variables-------------------------
        int grid = 1024, i, j, lp=1,max = grid, sizeGrid= grid*grid;
        unsigned int par = 160000, loop=2000, sizePar = 2*par;
        float t_i=0.0, t_mc_h2d=0.0, t_mc_d2h=0.0, t_pl=0.0, ti=0.0, tmc_h2d=0.0, tpl=0.0;
        cudaEvent_t s_i, e_i, s_mc_h2d, e_mc_h2d, s_mc_d2h, e_mc_d2h, s_pl, e_pl;
        float *netH, *pH, *netD,  *pD;
        //___Time flags___
        cudaEventCreate(&s_i);
        cudaEventCreate(&e_i);
        cudaEventCreate(&s_mc_h2d);
        cudaEventCreate(&e_mc_h2d);
        cudaEventCreate(&s_mc_d2h);
```

```
    cudaEventCreate(&e_mc_d2h);
    cudaEventCreate(&s_pl);
    cudaEventCreate(&e_pl);
    //_________________
//--------------------------------------------------------------------

//--------------------Initializing data-------------------------
    //___start clock___.
    cudaEventRecord(s_i,0);

        //___CPU Memory allocation___
        netH =        (float*)malloc(sizeof(float)*sizeGrid);
        pH   =        (float*)malloc(sizeof(float)*sizePar);
        //_______________________________
        //___initializing grid___
        for(i=0;i< grid;i++)
            for(j=0;j< grid;j++)
                    netH[grid*i+j]=0.0;
        //_______________________________
        //___Random particle position___
        for( i = 0; i < sizePar; i++)
            pH[i]= ((float)rand()/(float)(RAND_MAX) * (float)(max-1));
        //_______________________________

    cudaEventRecord( e_i,0 );
    cudaEventSynchronize( e_i );
    cudaEventElapsedTime( &ti, s_i, e_i);
    //_______________________________
//--------------------------------------------------------------------
//--------------------GPU memory allocation for grid-------------------------
    //___start clock___.
    cudaEventRecord(s_mc_h2d,0);

        //___GPU memory allocation___
        cudaMalloc( (void **)&netD, sizeof(float)*sizeGrid);
        //_______________________________
        //___Data Transfer___
        cudaMemcpy(netD, netH, sizeGrid*(sizeof(float)),  cudaMemcpyHostToDevice);
        //____________________

    cudaEventRecord( e_mc_h2d,0 );
    cudaEventSynchronize( e_mc_h2d );
    cudaEventElapsedTime( &tmc_h2d, s_mc_h2d, e_mc_h2d);
    t_mc_h2d+=tmc_h2d; //calculating time
    //_______________________________
//-----------------------------------------------------------------------------

//--------------------Parallel implementation -------------------------
for(lp=1;lp<loop;lp++){
//___perticle data transfer___
    cudaEventRecord(s_mc_h2d,0);
        //___Allocating GPU memory___
        cudaMalloc( (void **)&pD, sizeof(float)*sizePar);
        //___Memory transfer from CPU to GPU___
        cudaMemcpy( pD,   pH,   sizePar*(sizeof(float)), cudaMemcpyHostToDevice);
    cudaEventRecord( e_mc_h2d,0 );
    cudaEventSynchronize( e_mc_h2d );
```

```
    cudaEventElapsedTime( &tmc_h2d, s_mc_h2d, e_mc_h2d);
//___Launching threads___
    cudaEventRecord( s_pl,0 );
        //___thread dimentions___
        dim3 dimBlock(192);
        dim3 dimGrid((par/192));
        //___kernel Launch___
        parMap<<<dimGrid, dimBlock>>>(pD, netD, grid);
    cudaEventRecord( e_pl,0 );
    cudaEventSynchronize( e_pl );
    cudaEventElapsedTime( &tpl, s_pl, e_pl);
//___Time keeing___
    t_i+=ti;
    t_mc_h2d+=tmc_h2d;
    t_pl+=tpl;
}

//___copy results from GPU to CPU___
    cudaEventRecord( s_mc_d2h,0 );
        cudaMemcpy(netH, netD, sizeof(float)*sizeGrid, cudaMemcpyDeviceToHost);
    cudaEventRecord( e_mc_d2h,0 );
    cudaEventSynchronize( e_mc_d2h );
    cudaEventElapsedTime( &t_mc_d2h, s_mc_d2h, e_mc_d2h);
//--------------------------------------------------------------------
//--------------------Saving result in file -------------------------
    //___Opening file___
    FILE *f = fopen("file.txt", "w");
    par*=loop;
    if (f == NULL){
        printf("Error opening file!\n");
        exit(1);
    }

    float avg= par/(max*max);

    for ( i = 0; i < sizeGrid; ++i){
        fprintf (f,"%f ",((netH[i])/avg))  ;
        if (i%grid==(grid-1))
            fprintf (f," \n" );
    }

    fclose(f);
//--------------------------------------------------------------------------

    printf("\n\nGrid size: \t\t%d \n particle:\t %d\n", grid,par);
    printf("\nInitialisation time:\t%f \n", t_i);
    printf("\nMemory Copy H 2 D:\t%f \n", t_mc_h2d);
    printf("\nMemory Copy D 2 H:\t%f \n", t_mc_d2h);
    printf("\nProcessing time:\t%f \n\n", t_pl);

    //___destroy events___
    cudaEventDestroy(s_i);
    cudaEventDestroy(e_i);
    cudaEventDestroy(s_mc_h2d);
    cudaEventDestroy(e_mc_h2d);
    cudaEventDestroy(s_pl);
    cudaEventDestroy(e_pl);
```

```
        cudaEventDestroy(s_mc_d2h);
        cudaEventDestroy(e_mc_d2h);

        // Free memory
        cudaFree(netD);
        cudaFree(pD);
        free(netH);
        free(pH);

        return 0;
}
```