HPC - Assignment 2
CUDA

David Kroukamp (536705)
and
Wade Pimenta (544147)

University of the Witwatersrand, Johannesburg, South Africa

May 21, 2015

Outline

Problem Description

1D heat diffusion is the diffusion of heat along an infinitely narrow pipe. Initially, the whole pipe is at a stable and fixed temperature; for clarity purposes, we set our pipes initial temperature to be zero. At the start (time 0), we set both ends to a specified temperature, which remains fixed through the computation. We then calculate how the temperatures change in the rest of the pipe over time. Mathematically, the problem is to solve a 1D differential equation representing heat diffusion:

$$\frac{\delta u}{\delta t} = \frac{\delta^2 u}{\delta x^2}$$

Problem Description (cont.)

Our approach is to discretize the problem space by representing U by a one-dimensional array and computing values for a sequence of discrete time steps. Assume we have the values for U at time step k in an array Uk, then for the next time step $k+1$ update the second array $Ukp1$ as

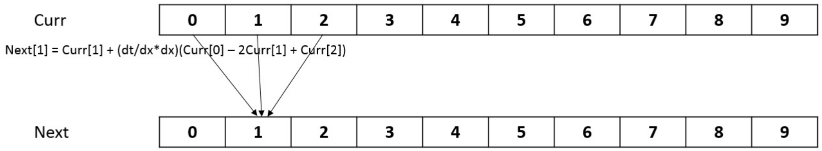
$$Ukp1[i] = Uk[i] + \frac{dt}{dx \times dx}(Uk[i+1] - 2Uk[i] + Uk[i-1])$$

where dt and dx represents the intervals between discrete time steps and between discrete points respectively.

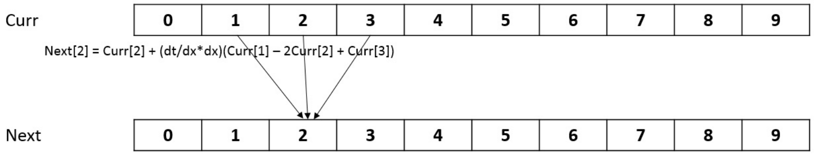
A Visual Representation

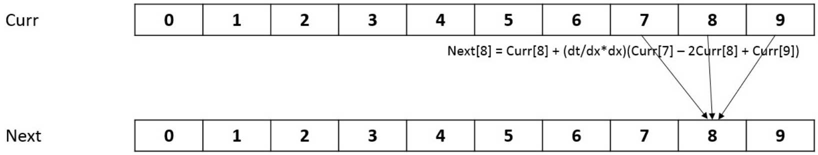
The following is a visual representation of the update function

$$Ukp1[i] = Uk[i] + \frac{dt}{dx \times dx}(Uk[i+1] - 2Uk[i] + Uk[i-1])$$


Curr
0
1
2
3
4
5
6
7
8
9
Next[1] = Curr[1] + (dt/dx*dx)(Curr[0] – 2Curr[1] + Curr[2])
Next
0
1
2
3
4
5
6
7
8
9

A Visual Representation (cont.)


Curr
0
1
2
3
4
5
6
7
8
9
Next[2] = Curr[2] + (dt/dx*dx)(Curr[1] – 2Curr[2] + Curr[3])
Next
0
1
2
3
4
5
6
7
8
9


Curr
0
1
2
3
4
5
6
7
8
9
Next[8] = Curr[8] + (dt/dx*dx)(Curr[7] – 2Curr[8] + Curr[9])
Next
0
1
2
3
4
5
6
7
8
9

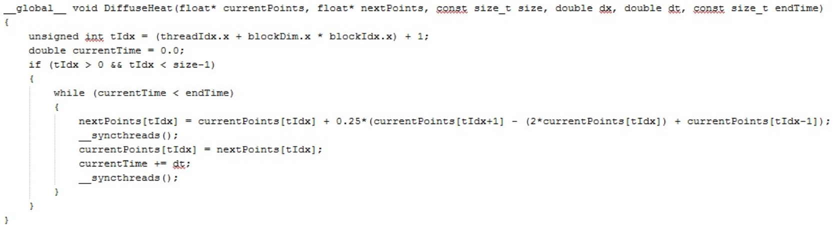
The CUDA Implementation

Visually, we can see that the calculation of each new element can be done at the same time as the other elements since there are no data dependencies.
We can apply the following algorithm:

```
if (tIdx >0 and tIdx <arraySize-1) then
    while currentTime <endTime do
        $nextArray[tIdx] \leftarrow$
$currentArray[tIdx] + 0.25(currentArray[tIdx - 1]$
$-2 * currentArray[tIdx] + currentArray[tIdx])$

Sync threads and copy contents of nextArray to currentArray
        Increment currentTime by timestep
    end while
end if
```

The Code



```
__global__ void DiffuseHeat(float* currentPoints, float* nextPoints, const size_t size, double dx, double dt, const size_t endTime)
{
    unsigned int tIdx = (threadIdx.x + blockDim.x * blockIdx.x) + 1;
    double currentTime = 0.0;
    if (tIdx > 0 && tIdx < size-1)
    {
        while (currentTime < endTime)
        {
            nextPoints[tIdx] = currentPoints[tIdx] + 0.25*(currentPoints[tIdx+1] - (2*currentPoints[tIdx]) + currentPoints[tIdx-1]);
            __syncthreads();
            currentPoints[tIdx] = nextPoints[tIdx];
            currentTime += dt;
            __syncthreads();
        }
    }
}
```