Dynamic Division Algorithm

Abstract:

In many GPU based arrays we would need to work with 3D arrays. While working with these arrays, we will need to slice them along primary dimensions to make our tasks easier. However, in case of large array dimensions it might be that we cannot even execute a single line as a thread. This is where DDA will come into effect, as an algorithm designed to automatically slice data independently of thread and block count. As a sample for the algorithm we will be running derivatives along all 3 dimensions.

Steps

1. Identify the primary size of the array, for example, we will assume it to be of size  with a uniform datatype. Length, Width and Height respectively.
2. We run the CUDA kernel as as Threads and Blocks respectively.
3. We know that the operation should run till index reaches from for a loop and then it will go over to the next line.
4. For the dimension, we can just use as our operation. We will extend this soon.

Execute as the basic kernel. Whenever index reaches X we reset index to 0 while bumping the grid count. Effectively, we calculate the derivatives of each point *without* counting the endpoints which will be handled by an *if* statement. 