CUDA Compute Inter-particle/atom Interactions Documentation

Rishiraj Chakraborty

February 2019

This documentation is intended to help read parts of the CUDA_SW_particle code, which compute inter-particle/atom interactions. The code itself is used to solve the Shallow Water Equations of fluid mechanics with dynamic particle tracking. One of the approaches in fluid mechanics (Lagrangian) uses particles to study the flow. To compute particle interactions, we need to measure inter-particle distances. The number of particles we typically use in our flows is pretty big, a million for example. The part of the code documented here uses buckets of specified threshold distance to compute interactions between particles in parallel using a GPU at a much cheaper cost than brute force.

The following sections describe the functions written in the corresponding files of CUDA_SW_particle responsible for carrying out the above mentioned computation. *interactions_grid*() is a cuda kernel and is called by the method *write_interactions*() which also gives a point-wise description of our algorithm.

cuda_kernels.cu function *interactions_grid*()

This is a CUDA kernel which adds pairs of interacting particles to the lists of interactions. The advantage of using CUDA is that, we can do the comparison for the particles in parallel.

Description of parameters:

- *part_pos_x*, *part_pos_y*: Pointers to spatial positions of particles sorted by their bucket indices.
- *grid_dim*: Pointer to the number of buckets in each direction.
- *bucket_begin*: Pointer to starting particle index in a bucket, length of the array being the number of buckets.
- *bucket_end*: Pointer to the last particle index in a bucket, length of the array being the number of buckets.
- *box_size*: Pointer to the dimensions of the box domain.
- *atom_index*: Pointer to particle indices.

- $flagstore$: Acts like a flag. Value 0 counts the number of interactions for each particle, and value 1 finds the interactive pairs.
- *num_interactions*: Pointer which stores the number of interactions for each particle when $flagstore = 0$.
- *start_interactions*: Used when $flagstore = 1$. This stores cumulative indices of the *num_interactions* array. This is used for determining the starting index of each particle in the storing lists $storepair1$ and $storepair2$
- r: This is the dimension of each bucket or the threshold interaction distance between particles.
- *num_particles*: Stores the number of particles used.
- $storepair1$, $storepair2$: Stores the interacting particle pairs respectively.

Description of variables:

- gx, gy: These convert the spatial positional values of a particle into grid index of the bucket it belongs in.
- $xshift$, $yshift$: These are used to access the neighbouring buckets of bucket.
- $gxshift$, $gyshift$: Stores the grid indices of adjacent buckets.
- $jbin$: The indices introduced above are directional and 2-D. We want to represent all the buckets using a 1-D index; $jbin$ stores the 1-D index of each bin.

Shallow_Water_Eq.cu function $writeInteractions()$

This function in Shallow_Water_Eq.cu is responsible for finding the interacting pairs of particles by using the *interactions_grid*$(**)$ CUDA kernel described above.

Description of variables

- *h_box_size*, *d_box_size*: Represents the size of the domain for the host and the device respectively.
- *h_grid_dim*, *d_grid_dim*: Tensors with number of dimensions same as that of the domain, with each dimension storing the number of buckets in that dimension for host and device respectively.
- *h_storepair*$*$, *d_storepair*$*$: Lists for storing the pairs of interactive particles for host and device respectively.

- *nx_intgrid*, *ny_intgrid*: Stores the number of buckets in the x and y directions respectively.
- *gs_x*, *gs_y*: Stores the grid spacing in the x and y directions respectively.

Description of algorithm:

1. We allocate indices to the particles from 0 to *num_particles*.
2. We find the bucket grid index for each particle.
3. We sort the particle positions based on the bucket grid indices in ascending order.
4. We store the first and last particle in each bucket in *bucket_begin* and *bucket_end* respectively.
5. We call *interactions_grid*($**$) once, to calculate the number of interactions (*num_interactions*) of each particle.
6. We calculate the starting position of each particle in the storing list by performing an exclusive prefix sum operation in *start_interactions*
7. We calculate the size of our storing lists by summing the last values in *num_interactions* and *start_interactions*.
8. We call *interactions_grid*($**$) again, but this time to calculate and store the interacting pairs of particles in the storing lists.
9. We sort the storing lists in ascending order of first particles in the pairs.