GROMACS - Task #2822

Make nbnxn a module

01/03/2019 02:05 PM - Berk Hess

Status:	Accepted	
Priority:	Normal	
Assignee:	Berk Hess	
Category:	core library	
Target version:	future	
Difficulty:	uncategorized	

Description

The nbnxn non-bonded functionality should be a proper module. The rest of Gromacs only needs access to a handle to call a few methods on.
TODO:
Put all nbnxn related code in an nbnxn directory. CUDA and OpenCL in two sub-directories, simd_4xn and simd_2xnn also in two sub-directories. Should the plain-C kernels be in a sub-directory as well?
Make a pimpled Nbnxn class. Only methods are a gridding call, grid order extraction (for DD), a make-pairlist call and non-bonded execute/launch call and output copyback/weight calls.

History

#1 - 01/03/2019 02:44 PM - Erik Lindahl

I think this needs to be split into several smaller and fully independent modules that are not aware of each other's internal data.

- On the lower level we have the concept of the cluster-list (or whatever descriptive name we should give it). This should not be concerned with calculating any interactions or domain decomposition, but simply a class that allows for different I vs. J cluster sizes, creating and manipulating lists, and querying their properties. My hunch is that it should also be fully independent of hardware - certain types of hardware might need a certain list setup, but that does not imply the fundamental list data structures must be aware of the hardware.
- The cluster-list-generation (neighborsearching) should be a separate module. It will only operate on the public methods of a clusterlist class (which in turn will have subclasses), but here we might need to create different types of geometries based on what the user requested (which in turn might depend on hardware, but the class itself likely does not have to be hardware-aware).
- Then we might need some special code for DD/communication. I think we'll get much simpler code by extracting that both from the clusterlist and neighborserarching and keeping it either as a separate module, or as part of some communication module.
- Finally, we have the code that calculates short-range nonbonded interactions given a set of coordinates and a cluster-neighborlist. This should only depend on the coordinates and fundamental list type, but neither how the list was generated (neighborsearching) nor how the data got to each node (DD/communication).

The easiest module to start designing is likely the clusterlist, since it doesn't have to depend on the others. Should we try a brainstorming session e.g. tomorrow?

#2 - 01/03/2019 03:07 PM - Berk Hess

Or course. Things like the grid will be it's own class with private data. The search only needs read access to the clusters on the grid.
The cluster list is only needed by the search code and the kernels, nothing else interacts with it (apart from that there is a a local and non-local part).
The grid is the only thing that interacts with the domain decomposition. The kernels don't need to know about this.

Note all this access is already very much separated in the code. This is only a matter of splitting up the current mess of include files and making all data private.

#3 - 12/18/2019 10:49 AM - Joe Jordan

- Target version changed from 2020 to future