# Axel’s email

I have had a quick look at your github repository and have some questions about your code:

Am I correct that the purpose of "compute position" is to collect and broadcast the distributed position (and atom ID and group mask) information from the individual processors to all processors? this runs contrary to the general parallelization strategy of LAMMPS and thus will inherently be serializing the data that was previously be distributed and worked on in parallel

Consequently, I don't see any attempt to parallelize the calculation in fix ordern and specifically it seems that all MPI ranks will open and write to the same file which can easily lead to file corruption on networked file systems. is there a specific reason that the data cannot be processed in parallel and with domain decomposition?

The behavior of compute rdf/ext runs contrary to the way how fixes and computes operate in LAMMPS. it operates like a fix, but being a compute it needs a "consumer" of a dummy property that the compute provides. also, I am not quite certain what is referred to in the original compute rdf not being corrected for finite size effects. its current implementation does consider the excluded pair count and thus the limit of g(r) for r -> oo is exactly 1.0 (and not n/(n-1)) for all cases that I have tested. is this perhaps referring to an older version of compute rdf (pre-summer 2017)?

Also there doesn't seem to be any protection from running with large systems (more than ~400M atoms) and all counters are implemented in 32-bit signed integers while LAMMPS uses for all counters referring to the entire system with 64-bit signed integers (unless compiled with -DLAMMPS\_SMALLSMALL).

Overall, this would be a contribution that needs to explicitly warn about the growing memory use and performance impact of applying it for larger systems, since it will cancel parallelization and data distribution. It would also be needed to convert the compute rdf/ext style to become a fix or use the same mechanism to access its output like the existing compute rdf instead of implementing its own mechanism.

we should also discuss the names of the classes and styles, since in my opinion they don't represent well what they do. This is important with a package as large as LAMMPS with so many features so that users will intuitively recognize that those may be useful computes/fixes for them.

# Tasks to Do/Check:

1. Change the definition of groups and how diffusion coefficients are computed
   * Remove the internal section in OCTP on grouping all atoms again (fix ordern, lines 492-532)
   * Pass the name of the generic LAMMPS groups, which will be passed to “compute position”, “fix ordern” and “fix rdf/ext” (instead of only “all” for all groups)
   * Preallocate the size of dynamic arrays based on the number of the specified groups
   * For the time being, diffusion calculations remain the same as on one core, but later we may need to change the code according to results of the last bullet
2. Does the original “fix rdf” compute the corrected RDF (versions after 2017)?
   * Check if fix rdf yields the correct RDF for a system of two atoms in a box
3. Can “compute rdf/ext” be parallelized better as it’s currently embarrassingly parallel (n\*(n-1) interactions)?
   * Should we also change “compute rdf/ext” to “fix rdf/ext”? If so what are the requirements to do so?
4. How is it possible to update the output files on a network without causing any problem? I think that I used a similar method for updating file as “fix” commands
5. Diminutive things to do
   * Change the variable assignments to 64-bit numbers (mainly integers)
   * Give a warning in case of a large number of atoms (note that the number of tracked atoms is important and not the number of atoms in the whole simulations box; tracking 100 atoms in a simulation of 100M atoms is not time consuming):
     + For a system of how many atoms, the system should give a warning? (~100k)
     + For a system of how many atoms, the system should give an error? (~1M)
   * Rename the fix and compute names to give a better giving the idea behind them
6. (optional) Verify if it is possible to decompose the computations on different processors (see my comment #A on Axel’s email):
   * For computing self-diffusion coefficients
   * For computing Onsager coefficients
   * For computing RDFs within a predefined cutoff radius