At certain MD steps (depending mainly on the user input), the PP ranks need to coordinate accumulating global quantities. The current implementation encodes logic from basically everything mdrun can do that might need to do that. It fills an array of doubles of minimal size for the present step to use with an MPI reduction, which sounds efficient, but the volume communicated is always small and the performance cost of the branching to decide what to fill it with is probably noticeable in some cases. Also, it is very difficult to understand and debug.

A better model might be to have modules that might want to contribute to such accumulation to do so by collaborating to prepare a fixed-size buffer (minimal for the present simulation, but perhaps excessive for the present MD step) that they fill when suitable, notify another object to manage the reduction, and later read from the buffer. This lets the module handle its own logic and check its own pre- and post-conditions.

This will reduce the complexity directly expressed in the code of do_md, because the control logic will live in the modules, and all it needs to do is call the reduction manager, which will already know if work is required of it. This will be necessary as we move towards a modular approach to implementing integrators.

In discussion with Pascal, we thought about whether it is OK that clients of global reduction only get notified of the fact that they should now have valid results in debug mode. We concluded that this is OK. Possibly we don't even need the notification in debug mode - see below. Key consideration here is that it's relatively cheap to write to memory that's already in cache, whereas a function call to a different translation unit (including to set data there) is relatively expensive.

If a client wants a stronger guarantee, then they can arrange for an acknowledgement be returned at a later stage (e.g. master notices that it is time to checkpoint, sets a signal, at next reduction step other ranks see that, send an acknowledgement to all, at next reduction step all can see the acknowledgement, so all can coordinate activity to be ready to write the checkpoint at the agreed step), or via an extra value in the accumulation payload (e.g. for lincs constraint violation, it would be fine, e.g. in debug mode to ask for three fields to be reduced, two with the real data, and one with the value one contributed by each rank, so that all ranks can compare this with the number of PP ranks and conclude that the rest of the fields have been accumulated).