GROMACS - Feature #2015

Auto recovery from dd communication error

07/26/2016 10:37 AM - Semen Yesylevskyy

Status:	New	
Priority:	Normal	
Assignee:		
Category:		
Target version:		
Difficulty:	uncategorized	

Description

One of the most nasty things in complex simulations is dd communication error:

"N particles communicated to PME rank N are more than 2/3 times the cut-off out of the domain decomposition cell..."

In most systems this error is not actually an error. It does not happen on single processor and is recoverable in parallel runs - if one restarts from checkpoint the system goes through the point of crash happily and runs for long time until this error happens again much later. As a result one have to restart simulations again and again which is very inconvenient.

If I understand the source of this error correctly it happens when dd algorithm is unable to adjust the cells to accommodate moving atoms but if dd is initialized from scratch upon restart the problem goes away. I propose to introduce a flag for mdrun which does auto recovery from such errors by resetting dd once the problem occurs. Mdrun will still crash by default but if the flag is set it will only print warning and try to auto-recover. If the problem occurs very often (an interval could be adjustable) it may still crash with error.

Such auto-recovery will simplify long simulations dramatically since the user would not be forced to manually resubmit the job each dew hours.

History

#1 - 07/26/2016 12:02 PM - Mark Abraham

Semen Yesylevskyy wrote:

> One of the most nasty things in complex simulations is dd communication error:
>
> "N particles communicated to PME rank N are more than 2/3 times the cut-off out of the domain decomposition cell..."
>
> In most systems this error is not actually an error. It does not happen on single processor and is recoverable in parallel runs - if one restarts from checkpoint the system goes through the point of crash happily and runs for long time until this error happens again much later. As a result one have to restart simulations again and again which is very inconvenient.

Indeed, we understand the frustration. However the implementation seems to work well for large numbers of kinds of simulations from multiple users, so from the point of view of the developer, the question is "why, if your system is well equilibrated, do particles sometimes seem to move quite far during the lifetime of a domain decomposition?" Attaching a sample .tpr +.cpt that once produced such a scenario would probably be helpful (along with some description of what's being simulated and how).

> If I understand the source of this error correctly it happens when dd algorithm is unable to adjust the cells to accommodate moving atoms but if dd is initialized from scratch upon restart the problem goes away.

Kind of. This code is managing the coordination between the real-space DD and the decomposition of real space across the PME ranks. In each decomposition, exactly one rank must have responsibility for each atom, but particles will move There's plenty of room in the implementation for the possibility that an edge case isn't being handled as well as it might be.

> I propose to introduce a flag for mdrun which does auto recovery from such errors by resetting dd once the problem occurs. Mdrun will still crash by default but if the flag is set it will only print warning and try to auto-recover. If the problem occurs very often (an interval could be adjustable) it may still crash with error.

Maybe. You can emulate this by keeping the lifetime of a DD short, ie by controlling nstlist (since the lifetime of the short-range neighbourlist is the same as that of the DD) e.g. with gmx mdrun -nstlist

> Such auto-recovery will simplify long simulations dramatically since the user would not be forced to manually resubmit the job each dew hours.

Your restart command line can have the same form whether the run crashed or not. What's the work you're doing manually?

#2 - 07/26/2016 12:19 PM - Semen Yesylevskyy

> Attaching a sample .tpr +.cpt that once produced such a scenario would probably be helpful (along with some description of what's being simulated and how).

I've submitted bug report about this issue already. In ideal world I'd wait until it is fixed but we have to do the job now. This is rather common things - we know that our setup crashed once per ~10 ns due to some corner cases but we still have to use it. It would be nice to provide the user with possibility to workaround such instabilities if he knows what he is doing.

> Your restart command line can have the same form whether the run crashed or not. What's the work you're doing manually?

If it crashes you have to resubmit the job. This is manual operation because most clusters nowadays prohibit auto-resubmitting scripts. If it recovers automatically the job is not killed and you have to do nothing.

> you can emulate this by keeping the lifetime of a DD short, ie by controlling nstlist

We are forced to use group cutoff (another bug, which is apparently already fixed) so nstlist doesn't always help.

#3 - 07/26/2016 01:43 PM - Mark Abraham

Semen Yesylevskyy wrote:

> > Attaching a sample .tpr +.cpt that once produced such a scenario would probably be helpful (along with some description of what's being simulated and how).
>
> I've submitted bug report about this issue already.

Which one? :-) We can't go around guessing which things are related, and have lots of users potentially reporting related issues, some of them are already fixed, etc.

> In ideal world I'd wait until it is fixed but we have to do the job now. This is rather common things - we know that our setup crashed once per ~10 ns due to some corner cases but we still have to use it. It would be nice to provide the user with possibility to workaround such instabilities if he knows what he is doing.

Yeah, but you're essentially proposing for the code to do an in-memory checkpoint every MD step, which requires a big re-design (there are reasons MC isn't implemented yet). If we'd done a better job in the first place, it'd be achievable, but for now, we can't.

> > Your restart command line can have the same form whether the run crashed or not. What's the work you're doing manually?
>
> If it crashes you have to resubmit the job. This is manual operation because most clusters nowadays prohibit auto-resubmitting scripts. If it recovers automatically the job is not killed and you have to do nothing.

Sure, but your job runs a shell script. That script doesn't have to be a bare call to mdrun. It can run a loop of mdrun, then gmx check on the checkpoint file, or inspect the end of the log file to see what progress was made, or why mdrun stopped, and perhaps ask the job scheduler how much time is remaining, and decide whether to restart. That's all a long way from ideal, but you can have it working tomorrow :-)

> > you can emulate this by keeping the lifetime of a DD short, ie by controlling nstlist
>
> We are forced to use group cutoff (another bug, which is apparently already fixed) so nstlist doesn't always help.

Sure it does... you can choose its value in the .mdp file. If your model physics is capable of moving particles in ways that make physical sense but which don't fit GROMACS design assumptions, then you might have to manage some details, and larger buffers or more frequent neighbour list updates are top of the list.

#4 - 07/26/2016 01:57 PM - Semen Yesylevskyy

> Which one? :-) We can't go around guessing which things are related, and have lots of users potentially reporting related issues, some of them are already fixed, etc.

As far as I remember if was related to #1958

> Yeah, but you're essentially proposing for the code to do an in-memory checkpoint every MD step, which requires >a big re-design (there are reasons MC isn't implemented yet). If we'd done a better job in the first place, >it'd be achievable, but for now, we can't.

I see. What about recovering from the latest checkpoint from the disk in such cases? I understand that this sounds a bit dumb since you can do this manually, but my point is that the user should not do anything manually if mdrun can fix its internal dd problem by its own. For me manual restart in such case looks like mdrun is telling me "Ok, I can't manage my internal data structures in a correct way, so I'll just spit an useless error and dye". Not very polite :)

> Sure, but your job runs a shell script. That script doesn't have to be a bare call to mdrun.

From the rules of our cluster: "you are not allowed to make more than one call of MPI program per job". I know that this is stupid, but this happens.

#5 - 07/26/2016 05:32 PM - Mark Abraham

Semen Yesylevskyy wrote:

> Which one? :-) We can't go around guessing which things are related, and have lots of users potentially reporting related issues, some of them are already fixed, etc.
>
> As far as I remember if was related to #1958

OK, well that's fixed with respect to the Verlet scheme in 5.1.3, so you have various new options.

> Yeah, but you're essentially proposing for the code to do an in-memory checkpoint every MD step, which requires >a big re-design (there are reasons MC isn't implemented yet). If we'd done a better job in the first place, >it'd be achievable, but for now, we can't.
>
> I see. What about recovering from the latest checkpoint from the disk in such cases?

It's not robust in general. You think you know enough that the restart can work (because any source of non-reproducibility is likely to avoid the problem behind #1958), but the error message is quite right that most of the time the reason the design assumptions behind the parallelism implementation have been violated is because the simulation is unstable to the point of being invalid. In the case of #9158, the issue was in the formulation of the Verlet kernels. So mdrun always attempting an automatic restart would be useless in all those cases, and perhaps even more confusing to the many new users who encounter them, even though it would probably fortuitously resolve #1958.

> I understand that this sounds a bit dumb since you can do this manually, but my point is that the user should not do anything manually if mdrun can fix its internal dd problem by its own.

This isn't an "internal dd problem." Data has to get sent from a set of As to a set of Bs, following a pre-determined schedule. Efficiency requires sending as few messages as needed, and for them to be as short as possible. Particles in a useful simulation don't move more than the length of the short-range list during the lifetime of the list, so of course that fact might be used in determining an efficient schedule.

> For me manual restart in such case looks like mdrun is telling me "Ok, I can't manage my internal data structures in a correct way, so I'll just spit an useless error and dye". Not very polite :)

There are different viewpoints on "correct management of data structures." The obvious bulletproof solution is to broadcast all coordinates everywhere just in case someone doing a niche flavour of simulation occasionally has atoms go flying across their simulation in not-very-reproducible ways, so that they get some other "your simulation is exploding" message next MD step, but normal people won't want to pay for that extra communication time when they don't need it.

> Sure, but your job runs a shell script. That script doesn't have to be a bare call to mdrun.
>
> From the rules of our cluster: "you are not allowed to make more than one call of MPI program per job". I know that this is stupid, but this happens.

Indeed, quite restrictive. There are many kinds of multi-scale modelling workflows that run multiple MPI programs per job, whether successively or MPMD style. You could observe to them that you've asked the GROMACS developers to implement such an automatic-restart shell script inside their MPI program and they won't, so how about being practical rather than someone writing C++ code to hide the fact of working around such a rule. :-) If what they really want is to restrict automatic re-submission, then they should be doing sensible things like looking at the ancestor processes of the shell that's calling the submission command.

#6 - 07/26/2016 05:52 PM - Semen Yesylevskyy

Sure, you are right that this error message indicates some problem. However, I still can't understand why it runs normally after restart. If there are too large forces in the system it will blow up at approximally the same step all the time (and I have seen this many times for really wrong systems). Here we have another story - it crashes with dd error each 10 ns or so. After restart it goes through and I don't see anything wrong in the trajectory. It's very strage to see a crash after millions of sucesful steps...

#7 - 07/26/2016 07:24 PM - Roland Schulz

Which version of GROMACS are you using? If you aren't using 5.1.3 does it still blow up with that version? In a regular MD simulation you shouldn't get that error. One possible reason is that there is some software bug which causes it to blow up rarely. In that case the error message you get isn't the problem but a result of a much more important problem. If you still get the crash with 5.1.3, it would be good to know if you run with "mdrun -reprod -dlb no" whether after it crashes it crashes again at the same step after restarting. In that case we could debug why it crashes for you. That would let us fix the actual problem (to make sure you can trust your results) rather than hiding the problem.

#8 - 07/26/2016 10:02 PM - Mark Abraham

Semen Yesylevskyy wrote:

> Sure, you are right that this error message indicates some problem. However, I still can't understand why it runs normally after restart. If there are too large forces in the system it will blow up at approximally the same step all the time (and I have seen this many times for really wrong systems). Here we have another story - it crashes with dd error each 10 ns or so. After restart it goes through and I don't see anything wrong in the trajectory. It's very strage to see a crash after millions of sucesful steps...

If it's #1958, then you never had problems with the group scheme, and only had problems with the Verlet scheme, which were intermittent because the trigger was particles overlapping exactly (which won't tend to recur identically in a second run), and which we anyway think are fixed in 5.1.3. So there's actually no specific reason to consider implementing this feature. Please try 5.1.3 and let us know how you go :-)

#9 - 07/27/2016 08:26 AM - Semen Yesylevskyy

I'm using 5.1.2. The problem is observed with the cut-off scheme, so apparently it is different from #1958. I'll ask the administrators to install 5.1.3 and will try to see if it still occurs.

#10 - 07/28/2016 03:17 PM - Mark Abraham

Semen Yesylevskyy wrote:

> I'm using 5.1.2. The problem is observed with the cut-off scheme, so apparently it is different from #1958. I'll ask the administrators to install 5.1.3 and will try to see if it still occurs.

OK, but please see also my comments on #1965