GROMACS - Bug #2136

mdrun -noconfout checkpointing issue

03/08/2017 08:40 PM - Berk Hess

Status:	Closed		
Priority:	Normal		
Assignee:			
Category:	mdrun		
Target version:			
Affected version - extra info:		**Difficulty:**	uncategorized
Affected version:	2016		

Description

mdrun -noconfout does not write confout.gro and a checkpoint at the end of the run. This is useful for benchmarking, where significant time can be spent in writing confout.gro (although I don't know how much time is spent in writing state.cpt.
But for running in queuing systems it can be useful to suppress confout.gro but still write state.cpt. I think we should always write state.cpt at the end of a run. Maybe we should add a benchmarking option.

Related issues:

Related to GROMACS - Task #1781: re-design benchmarking functionality	**Accepted**
Related to GROMACS - Feature #2224: Proposed feature: conditional stop	**New**
Related to GROMACS - Bug #2377: Useless spam of "No option -multi"	**Closed**

Associated revisions

Revision f5cb6c13 - 01/11/2018 01:41 AM - Mark Abraham

Announce in user log files that features are deprecated.

These are merely informational notes, not warnings or errors.

Refs #1781, #1971, #2136

Change-Id: I96e19acb0e15d3f42b0929f555b451299a2882e4

History

#1 - 03/09/2017 05:16 PM - Mark Abraham

I don't think the time spent actually writing any file is relevant, but rather that it requires an allreduce before the (wasted) final update stage, and that is recorded as part of the run time, and that implementation would be exceedingly painful to separate, so you want to just suppress it all during benchmarking.

A hidden option for "write final output files" (perhaps yes/no/auto with auto meaning no if any other benchmarking-focussed option is being used) might work. If so, then mdrun -confout doesn't need to exist.

I agree that normal runs should always write a final checkpoint, and default to writing the confout file (if we keep mdrun -confout).

#2 - 04/22/2017 10:34 AM - Erik Lindahl

-noconfout sounds like a great candidate for an option to just kill.

We need to learn that the question is not how much can be added, but what can be taken away without crippling normal usage. Not having this option isn't going to hurt any normal benchmarks, and it means we have one less thing to worry about or test (which we obviously haven't been doing in this case).

Introducing an even more complex hidden option would just complicate things even further. Kill, kill, kill :-)

#3 - 04/22/2017 12:34 PM - Mark Abraham

You should have heard the screaming when I proposed removing mdrun -nsteps x in favour of mdrun -s x.tpr

#4 - 04/22/2017 12:44 PM - Mark Abraham

- Related to Bug #182: grompp is called with the -np N option. added

#5 - 04/22/2017 12:45 PM - Mark Abraham

- Related to Task #1781: re-design benchmarking functionality added

#6 - 04/22/2017 12:45 PM - Mark Abraham

- Related to deleted (Bug #182: grompp is called with the -np N option.)

#7 - 04/25/2017 09:39 AM - Berk Hess

My request was rather the other way around. I'd rather not have a confout.gro at all. What's the point of having a separate, bulky formatted (20x as large as xtc), inaccurate configuration file only for the step that happens to be the last output frame? You should have the information in a trajectory file already. This just adds to the clutter of output files and mdrun options.
Getting rid of confout.gro solves many issues at the time.

#8 - 04/25/2017 09:53 AM - Mark Abraham

Berk Hess wrote:

> My request was rather the other way around. I'd rather not have a confout.gro at all. What's the point of having a separate, bulky formatted (20x as large as xtc), inaccurate configuration file only for the step that happens to be the last output frame? You should have the information in a trajectory file already. This just adds to the clutter of output files and mdrun options.
> Getting rid of confout.gro solves many issues at the time.

People use it for grompp input for their next stage, so we need to provide a deprecation period and a suggested alternative. And perhaps ask for updates to tutorials.

#9 - 08/02/2017 09:59 PM - Roland Schulz

- Related to Feature #2224: Proposed feature: conditional stop added

#10 - 12/12/2017 11:09 PM - Erik Lindahl

If somebody wants to keep the noconfout option, this needs urgent attention by that person :-)

Otherwise I'll create a change to remove the option in a day or two.

#11 - 12/14/2017 02:45 AM - Mark Abraham

Removing -noconfout will mean that benchmarking results will include the cost of final reduction for cpt and gro output. I'm OK with that, but that particular option is not causing problems that are urgent to solve. I note that the noise of crickets has dominated #1781 for 9 months, while those who want the performance measurement features behind gmx mdrun -nsteps -resetstep -noconfout have been developing performance features and optimizations. I threatened to remove similar command-line options at https://redmine.gromacs.org/issues/1781#note-38 and while we had some useful discussion on the design requirements, nobody made concrete proposals for reforming the implementations for such features when I asked for them. So maybe we just have to remove it all in January 2019 and start fresh so that people are forced to engage in design and implementation. Make all of them deprecated in 2018 release?

I don't think there's an easy way of stopping data collection, nor coordinating that with PME-only ranks, even if we would try to arrange to stop timing regions so that the code in finish_run() might get run early and then we ignore the subsequent time. But if we could, then -noconfout stops being desirable for benchmarking.

Removing the gmx mdrun -c confout.gro output would require that gmx grompp -f new.mdp -c original.gro -t state.cpt always works regardless of what gets changed in the .mdp file. People seem to do gmx grompp -f new.mdp -c confout.gro during system preparation stages, because they can. Making gmx grompp -t compulsory (and accepting any coordinate file, because a .cpt is not available for the initial phase) would serve to educate users that we need a source of names (ie. -c, which could be a gro/etc. or old .tpr) and a source of state, including coordinates (ie. -t which could be a gro/etc. or a cpt). That would then let us change mdrun to stop writing confout.gro. But we can't do any of that for release-2018.

#12 - 12/26/2017 12:02 AM - Erik Lindahl

In the checkpointing code we have clearly said that checkpointing is ONLY intended for restarting runs with exactly the same version of GROMACS, and that any other use is really an abuse of the feature (and for that reason there is nowhere near the amount of checks that would be required for general IO, and no backward compatibility whatsoever. The CPT file format itself also completely lacks documentation.

IMHo, that is not a candidate to use for our general coordinate input to start simulations, since it's perfectly reasonable to even want to start a run in an older version of GROMACS from newer input data without asking the user to first compile a newer version of Gromacs so they can extract files that are possible to use with their version.

At some point it would be great to have a strictly defined file describing the physical state of a system (but NB: then it cannot change frequently). However, until we have that my vote is for avoiding the complexity of even more options - we all know that no matter how well intended, it results in bugs.

#13 - 12/26/2017 12:16 AM - Mark Abraham

Erik Lindahl wrote:

> In the checkpointing code we have clearly said that checkpointing is ONLY intended for restarting runs with exactly the same version of GROMACS, and that any other use is really an abuse of the feature (and for that reason there is nowhere near the amount of checks that would be required for general IO, and no backward compatibility whatsoever. The CPT file format itself also completely lacks documentation.
>
> IMHo, that is not a candidate to use for our general coordinate input to start simulations, since it's perfectly reasonable to even want to start a run in an older version of GROMACS from newer input data without asking the user to first compile a newer version of Gromacs so they can extract files that are possible to use with their version.
>
> At some point it would be great to have a strictly defined file describing the physical state of a system (but NB: then it cannot change frequently). However, until we have that my vote is for avoiding the complexity of even more options - we all know that no matter how well intended, it results in bugs.

Good point. Then we should not remove -c because then there is no readily usable way to access the end point of a simulation that is portable across versions, even at not-full precision.

#14 - 12/26/2017 01:15 AM - Mark Abraham

- Status changed from Accepted to Resolved

I think we have resolved the we should do nothing. We need to keep the -cpo and -c output as they are. Suppressing -c is a niche use (either you are benchmarking or you know you aren't going to use the file). Benchmarking features should be properly designed to have no impact on normal users. Writing -c after writing -cpo incurs almost no additional overhead, and users who don't want the file can use rm in their script.

-confout should go, however, so people are on notice that we'll remove it for 2019

#15 - 12/29/2017 11:18 AM - Erik Lindahl

- Status changed from Resolved to Closed

#16 - 01/03/2018 08:25 PM - Szilárd Páll

> -confout should go, however, so people are on notice that we'll remove it for 2019

Why not also stop dumping a final .gro, it would surely simplify code and the last frame can trivially be obtained form the trajectory, can't it?

#17 - 01/03/2018 08:50 PM - Mark Abraham

Szilárd Páll wrote:

> > -confout should go, however, so people are on notice that we'll remove it for 2019
>
> Why not also stop dumping a final .gro, it would surely simplify code and the last frame can trivially be obtained form the trajectory, can't it?

It can. But see my comment 8. If we make it harder to use the code, what is the benefit?

#18 - 01/03/2018 09:03 PM - Szilárd Páll

Mark Abraham wrote:

> It can. But see my comment 8. If we make it harder to use the code, what is the benefit?

Reduced code complexity. confout.gro is redundant, the data is both in the trajectory and cpt file, isn't it? It also serves only a subset of the users/use-cases and its result can be replicated with minimal effort through the invocation of a one-liner gmx subcommand.

#19 - 01/04/2018 09:49 AM - Erik Lindahl

As already explained in the thread, cpt is an undocumented format that keeps changing and where we do not guarantee that old formats can be read, and no normal user writes full precision trajectories (in particular since we recommend against it for all cases but velocities due to size). The concept of just running a command assumes that one has saved the original TPR file, and either that a GROMACS version 10 years from now can still read it, or that the old version still compiles.

At the end of the day, the interests of the large "subset" of users just trying to use the program without being experts outweigh a slight convenience for a handful of people who would like to get accurate benchmarks with shorter runs.

#20 - 01/04/2018 10:22 AM - Mark Abraham

Szilárd Páll wrote:

> Mark Abraham wrote:
>
> > It can. But see my comment 8. If we make it harder to use the code, what is the benefit?
>
> Reduced code complexity.

There's 50 lines in trajectory_writing.cpp, a fair bit of which is working around other designs with issues (single-rank runs re-using state_global; periodic molecule support)

> confout.gro is redundant, the data is both in the trajectory and cpt file, isn't it?

No, the trajectory often only has coordinates, and the cpt doesn't have names (so can't be used for gmx grompp -c). See my comment 11. Many user workflows I've seen on gmx-users (probably from them following tutorials) currently use the mdrun -c output as input for e.g. the next stage of equilibration for grompp -c. One might use gmx grompp -c old.gro -t new.cpt/new.trr but we'd first have to assess whether that use of -t is correct with respect to likely changes of mdp options, such as coupling algorithms, e.g. no fatal error because stuff is missing from the cpt, or unexpectedly found in the cpt. We also don't want to paint ourselves into a corner of having to be able to read all old cpt formats.

> It also serves only a subset of the users/use-cases and its result can be replicated with minimal effort through the invocation of a one-liner gmx subcommand.

But when I proposed removing mdrun -nsteps 1000 in #1781 because gmx convert-tpr -o steps1000 -nsteps 1000; gmx mdrun -s steps1000 was low effort, I got the impression you thought that that change in workflow was too large an effort for developers? In each case, our proper approach should prioritise making the code easy for users to use correctly, and easy for us to implement and maintain correctly over time.

#21 - 01/10/2018 10:32 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2136.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~release-2018~I96e19acb0e15d3f42b0929f555b451299a2882e4
Gerrit URL: https://gerrit.gromacs.org/7451

#22 - 01/11/2018 03:25 PM - Aleksei Iupinov

- Related to Bug #2377: Useless spam of "No option -multi" added