GROMACS - Task #1971

Removing buggy features vs. keeping workflows

05/26/2016 06:30 PM - Erik Lindahl

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

Description

For quite a few recent bugs we have ended up in this situation:

1) For convenience, we have added hacks or extra command line options to enable some workflow or save time.
2) These hacks suddenly mean there are half-a-dozen ways to do things (e.g. continue simulations), which leads to bugs.
3) The person using the workflow doesn't care too much, because they are not hit by the bug.
4) When somebody else tries to clean it up by removing features, there are (understandable) complaints that we should not break workflows.

Most of us think it is more fun to work on performance and features rather than stability/usability/fixing, which tends to compound the problem so these issues stay partially broken for years (one current example is the -deffnm feature).
This makes it extremely unthankful for the people who do try to fix other developers' bugs (in particular Mark) since they constantly get complaints that features cannot be removed, but very little help with the actual fixing.

So, for release-2017 and the future I would suggest to change the rules of that game a bit:

1) It should still be perfectly OK to say that a feature is important and that it should stay, but that has to come with with an implied promise to help fix the bugs it causes.
2) In that case, people who want it removed because it is buggy should hold off for a few months (or maybe even one full release).
3) However, a large part of the responsibility for fixing things fall on the person(s) who wanted the feature to stay.
4) When no fixing happens (which this far has been the norm), the "important workflow" argument should no longer carry any significant weight.

Mark had a good formulation during the Göttingen workshop that we (and other projects) tend to build up a technical "debt" when we have added code and features that is not as foolproof as it should be. At some point we need to pay that debt, either in terms of no longer having features that are buggy (no matter how useful a workflow might be to an expert user not hit by the bug), or by helping to fix them.

Related issues:

Related to GROMACS - Bug #1117: ensemble-averaged distance restraints is prob...	**Closed**	
Related to GROMACS - Feature #1292: mdrun features to deprecate for 5.0	**Closed**	**01/20/2014**
Related to GROMACS - Task #1323: determine future of existing tools for	**New**	
Related to GROMACS - Feature #1500: Post-5.0 feature clean-up plan	**New**	**01/20/2014**
Related to GROMACS - Bug #1998: membed is apparently broken	**Closed**	
Related to GROMACS - Bug #1354: Constant acceleration NEMD is broken.	**New**	
Related to GROMACS - Feature #753: Use of GB in parallel and/or with all-vs-a...	**Closed**	
Related to GROMACS - Feature #1095: Fix all-vs-all kernels	**Closed**	
Related to GROMACS - Task #1781: re-design benchmarking functionality	**Accepted**	
Related to GROMACS - Feature #2224: Proposed feature: conditional stop	**New**	
Related to GROMACS - Task #2391: re-enable TPI test	**Closed**	
Related to GROMACS - Bug #1249: no-PBC no-cutoff is broken with SIMD group ke...	**Closed**	**05/14/2013**
Related to GROMACS - Task #1852: Remove group scheme	**New**	

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

Revision 6a334106 - 01/23/2018 05:51 PM - Mark Abraham

Remove topology support for implicit solvation

Refs #1500
Refs #1971

Change-Id: I75d05d52ea1d528f63f2249da62f0bcfca4274d2

Revision 93c6b6d6 - 01/23/2018 06:08 PM - Mark Abraham

Remove support for implicit solvation

Mdp files with implicit-solvent = no can still be read, and formerly valid related fields are now ignored, so that default mdp files from previous versions of GROMACS will work. Anything else for the implciit-solvent mdp value gives an error in grompp.

grompp can now only write a tpr file that has a false value for ir->implicit_solvent, but can read older versions. When mdrun is presented with an older tpr file that did such a simulation, it refuses to run, presenting a useful error message. Such tpr files are still useful for other purposes, so can still be read, except that the fields specific to these methods are ignored.

grompp now ignores the topology directives for related parameters, which means that force-field folders that are the same as, or modifications of folders formerly supported by GROMACS still work. However, the versions currently distributed have none of those fields.

The group-scheme kernels have been removed, and generation infrastructure updated so that they do generate the code that's in the repo. However, now that the python generation scripts no longer generate GB kernels, the dictionary ordering changes, which changes the generated output. That output is not sensitive to the order of the declarations or data-structure elements, so this is only a cosmetic issue.

Documentation has been removed.

Unit tests on .mdp file handling have had to be updated.

Also removed unused enbcoul enumeration

Refs #1500
Refs #1971
Fixes #1054

Change-Id: Ib241555ff3d8e60012ba0e628ab0f9a3f91eca9e

History

#1 - 05/26/2016 07:07 PM - Roland Schulz

It might make sense to ask the people who want it removed to provide a failing test. Then if the people who want to keep the feature don't fix the failing test in a reasonable amount of time, then it gets removed.

#2 - 05/26/2016 07:36 PM - Erik Lindahl

That might work in some cases where issues are more isolated. However, I think our largest problems are with old pieces of code that interact with tons of other options (e.g. all our different ways to restart/continue/modify trajectories). The challenge there isn't whether we can identify one problematic combination (that can rapidly be fixed with a hack), but that bad (old) design choices or too many options create highly bug-prone code.

At some point such code has to be either redesigned or removed - we should probably not try to fix it by adding even more hacks.

#3 - 05/27/2016 12:30 AM - Mark Abraham

Roland Schulz wrote:

> It might make sense to ask the people who want it removed to provide a failing test. Then if the people who want to keep the feature don't fix the failing test in a reasonable amount of time, then it gets removed.

That would be useful if one already has supporting material for a bug report. It's somewhat useful in the case that e.g. some code still works at limited parallelism (GB, probably ensemble-averaged distance restraints) but segfaults elsewhere... other cases would need the kind of work and analysis that could otherwise be feature development, or scientific work.

The alternative is somewhat easier in the (sadly) expected case that nobody else is going to do anything. Someone who states they are using a feature in an important workflow can produce at least a functional minimal set of grompp inputs in under an hour. But if person who can do that hasn't managed to do that in a few years, then clearly this is only an important workflow if it comes at the expense of someone else's time (e.g. core developer maintaining or fixing, or person answering user questions, or users who run into the combinatorial buggy cases).

See discussion at #1117, #1292, #1323, and #1500. In all cases, what progress there has been has been to remove some of the functionality in question, largely by me, and sometimes in the face of passive resistance (often people have no time to contribute to non-constructive outcomes). Features largely maintained by Carsten Kutzner (e.g. enhanced rotation, computational electrophysiology, essential dynamics) are a notable exception.

#4 - 05/27/2016 12:31 AM - Mark Abraham

- Related to Bug #1117: ensemble-averaged distance restraints is probably broken added

#5 - 05/27/2016 12:31 AM - Mark Abraham

- Related to Feature #1292: mdrun features to deprecate for 5.0 added

#6 - 05/27/2016 12:31 AM - Mark Abraham

- Related to Task #1323: determine future of existing tools for added

#7 - 05/27/2016 12:31 AM - Mark Abraham

- Related to Feature #1500: Post-5.0 feature clean-up plan added

#8 - 05/27/2016 12:37 AM - Peter Kasson

I would argue that there should be another category. If saying that a feature is important requires a commitment by that individual to maintain the code in the face of other people's changes, then Gromacs becomes defined as a code by and for developers. That's all well and good, but **if** we decide that it should be a code for some user base, there should be a category of "a lot of users who may not be developers depend on this". My $.02.

#9 - 05/27/2016 12:54 AM - Erik Lindahl

> I would argue that there should be another category. If saying that a feature is important requires a commitment by that individual to maintain the code in the face of other people's changes, then Gromacs becomes defined as a code by and for developers. That's all well and good, but if we decide that it should be a code for some user base, there should be a category of "a lot of users who may not be developers depend on this". My $.02.

I don't think it is any different. It doesn't necessarily require a commitment from the first person wanting it, but it will require a commitment from somebody doing development (new or old person) - since at the end of the day, somebody will have to actually fix the bugs, redesign, and maintain even that piece of code.

Thus, we're highly unlikely to remove pdb2gmx tomorrow. Even though the design is horrible, we and other people need it, so we do our best to maintain it while grinding our teeth. Some shiny day it will be redesigned.

However, there are no developers apart from developers. If nobody wants to spend time on something... by definition we're not spending time on it.

#10 - 05/27/2016 01:00 AM - Mark Abraham

Peter Kasson wrote:

> I would argue that there should be another category. If saying that a feature is important requires a commitment by that individual to maintain the code in the face of other people's changes, then Gromacs becomes defined as a code by and for developers. That's all well and good, but **if** we decide that it should be a code for some user base, there should be a category of "a lot of users who may not be developers depend on this". My $.02.

Yes, we do also make such choices in some cases (e.g. Berk implemented [intermolecular_interactions] and we searched for over a year for someone somewhere to alpha test it, but the element of choice for the developer community has to be there. GROMACS can be a community code, but that means those being part of the community have to contribute time in terms of code development/maintenance/removal, documentation, testing and community engagement. For example, I prioritise fixing issues for Justin Lemkul and Chris Neale, because of their long history of being good citizens on multiple fronts (and there are others in that category). But someone who isn't able to contribute to the community for a long time is going to find that their stock might devalue. :-) If e.g. Michael hadn't been at least somewhat able to respond to bug reports and my cleanup in VV code, it would have been gone years ago.

#11 - 05/31/2016 06:00 AM - Chris Neale

It occurs to me that if workflows are ever removed from further releases, then there may be a new set of cases for which some error is found in an old code base and the usual reply of "try updating to a newer version of gromacs" is somewhat more complicated because there are strong reasons for the user not to update (because the feature they need was removed). I'm not weighing in one way or another on the main discussion point here, just pointing out a possible side-effect. When some feature gets dropped for lack of developer interest (including users as possible developers, etc.) then it seems that there also an implicit decision to either fix bugs in ever older versions of gromacs or to define those versions as unsupported even when some people may be clinging to them for niche (or possibly non-niche) routines.

#12 - 05/31/2016 07:33 AM - Erik Lindahl

Yes, such versions will still sadly become unsupported. However, one should remember that this happens in commercial software all the time (anybody using Aperture on OS X had to change their workflow, and you can't run a 5-year old version of OS X on new hardware). The huge advantage with open source is that people still have a chance to help fix things (or convince other people to help them by being a great community citizen in other ways), and then they will stay there.

However, there's no such thing as a free lunch: some of these features used by just a handful of users cause major grief/bugs for others, or they complicate the code a lot. This might sound harsh, but the point isn't that somebody will make dictatorial decisions about what stays vs. goes. Every single developer has a vote in these matters - but that vote is implicitly cast when each of us decides not to work on fixing a broken feature that is badly implemented.

#13 - 06/30/2016 03:55 PM - Mark Abraham

- Related to Bug #1998: membed is apparently broken added

#14 - 07/11/2016 08:14 PM - Mark Abraham

- Related to Bug #1354: Constant acceleration NEMD is broken. added

#15 - 07/11/2016 08:45 PM - Mark Abraham

- Related to Feature #753: Use of GB in parallel and/or with all-vs-all kernels needs a mention in the manual added

#16 - 07/11/2016 09:11 PM - Mark Abraham

- Related to Feature #1095: Fix all-vs-all kernels added

#17 - 03/07/2017 07:20 PM - Mark Abraham

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

#18 - 03/07/2017 07:27 PM - Mark Abraham

Discussion has arisen about the usefulness of mdrun command-line options for benchmarking (#1781 particularly starting at comment 38), since we've now had several bugs with the way useful-looking things don't combine well with features that were not considered in combination. We should not hold ourselves to the same standards for things like mdrun -reset* and mdrun -nsteps as we do for mdrun -deffnm (for example). Forcing people to stop using an inferior workflow sooner can have constructive value to offset the costs of disruption (e.g. mdrun -tunepme -resetstep 1000 -nsteps 6000 could include part of the tuning period in the data collection period vs mdrun -tunepme -benchmarksteps 5000)

#19 - 08/02/2017 10:00 PM - Roland Schulz

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

#20 - 01/10/2018 10:32 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #1971.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~release-2018~I96e19acb0e15d3f42b0929f555b451299a2882e4
Gerrit URL: https://gerrit.gromacs.org/7451

#21 - 01/16/2018 07:08 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1971.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~master~Ib241555ff3d8e60012ba0e628ab0f9a3f91eca9e
Gerrit URL: https://gerrit.gromacs.org/7477

#22 - 01/19/2018 01:36 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1971.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~master~I75d05d52ea1d528f63f2249da62f0bcfca4274d2
Gerrit URL: https://gerrit.gromacs.org/7498

#23 - 01/23/2018 10:51 AM - Aleksei Iupinov

- Related to Task #2391: re-enable TPI test added

#24 - 03/06/2018 11:15 PM - Mark Abraham

- Related to Bug #1249: no-PBC no-cutoff is broken with SIMD group kernels in 4.6.1 added

#25 - 10/01/2018 08:38 AM - Mark Abraham

- Related to Task #1852: Remove group scheme added