GROMACS - Feature #1500

Feature # 1292 (Closed): mdrun features to deprecate for 5.0

Post-5.0 feature clean-up plan

05/13/2014 10:03 AM - Mark Abraham

<table>
<thead>
<tr>
<th>Status:</th>
<th>New</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Mark Abraham</td>
</tr>
<tr>
<td>Category:</td>
<td>core library</td>
</tr>
<tr>
<td>Target version:</td>
<td>Uncategorized</td>
</tr>
</tbody>
</table>

Description

General status summary:

- #1292 (removing features for 5.0) is complete as far as I could achieve it on my own. Thanks to those who did contribute. Main items that happened there are removal of PD, and clean up for planned removal of iterated code for MTTK constraints.
- The Verlet scheme will be close to feature-complete in 5.0. The main thing missing is support for user tables, which is one of the things Alfredo Metere is working on. Other minor issues are that the GB parallelization is probably broken for both group and cutoff (so until someone wants to put time into it again, we support GB after 5.0 only in all-vs-all mode), and adress is only implemented for the group scheme (which needs at least new kernels, which should be addressed after we produce a kernel-generator python script).
- We do want a release where the Verlet and group schemes are both as reliable as we can reasonably achieve, so that any issues of cross-validating protocols can be settled unambiguously. I see that as the main reason for having a 5.1 later this year. Many of the group-scheme protocols are not worth reproducing, however.

Suggested features to announce as deprecated in 5.0

- group scheme (grompp already announces this; remove after 5.1)
- twin-range multiple-time-step scheme (it's been broken since before 4.5 #1400, and my attempt to fix it lately has not clearly succeeded; remove after 5.0)
- Generalized Born other than all-vs-all (#1054; remove after 5.0)
- iteration scheme for MTTK with constraints (remove after 5.0)
- ensemble-averaged restraints (unless someone actually is using it in ~4.6 and has a setup showing that it still works - discussion at #1117; remove after 5.0)
- mdrun -testverlet (since Verlet will be the default and people should no longer be converting ancient .tpr files - discussion at #1424; remove after 5.0)
- heuristic updates for group scheme (complicates signalling, which currently hurts multi simulations and makes it hard to verify a solution for #692, and Berk hopes to implement a better version of this for the Verlet scheme; remove after 5.0)

I don't like breaking/removing things that work, but "keep it, it may once have worked" is not a reasonable barrier to progress for those of us working on things we need for the future. "Here, keep this working" can be reasonable! :-)
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 implicit-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: lb241555ff3d8e60012ba0e628ab0f9a3f91eca9e

History

#1 - 05/13/2014 10:36 AM - Mark Abraham
- Description updated

#2 - 05/13/2014 10:47 AM - Mark Abraham
- Description updated

#3 - 05/13/2014 10:48 AM - Mark Abraham
- Description updated

#4 - 05/13/2014 10:48 AM - Mark Abraham
- Description updated

#5 - 05/13/2014 11:57 AM - Justin Lemkul
Related to the removal of the group scheme, the current code requires the use of group for anything using shells/Drudes (fatal error in shellfc.c if not). My current Drude branch is close to being ready, and I was hoping to have it considered for 5.1, so I wanted to mention that it's very important for us to make sure the shell/Drude code is supported going forward. I don't know what all that will entail as far as the Verlet kernels, but I wanted to note it here.
Very happy to continue to support such functionality, and happy to consider the new stuff for 5.1, but we will need to have a plan to port things to the Verlet scheme, and execute it later this year.

Other proposed things to deprecate and remove:

- Description updated

sd2 integrator (temperature is known to be too high; noticed by various people at various times, connected with #1496 though no report is there. The new sd integrator at https://gerrit.gromacs.org/#/c/3419 does better and is faster, and that patches announces the deprecation)

Should we deprecate some older compilers? E.g. ICC<13, gcc<4.6, clang<3.3. Of course users can still use them but we wouldn't fix compiler warnings or have pre-submit Jenkins tests. It is a bit annoying to have to fix compiler warnings for ancient compilers: e.g. http://jenkins.gromacs.org/job/Gromacs_Gerrit_master/5667/

Sounds good. Also MSVC2010 (at least?), and xlc<12.

Its good idea. Anyway most clusters admins and users installs their own recent compiler versions.

And perhaps some warnings when using an unsupported compiler version shipped on a so-called "enterprise" distributions.

In that case enterprise are so enterprise so for example ~30-40% of ki cluster users run gentoo prefix in their accounts or build some recent gcc version in $HOME

More thoughts:

IMO, anything in contrib that looks like it might no longer work can go. If someone wants it, git knows about it, and if they do want it, either checking out the git version where they were committed, and/or doing some fix up is still going to be needed. I know I haven't been maintaining anything there during refactoring patches, though Teemu sometimes does.

Erik and I suggest that most of the analysis tools should move to contrib, to reflect their effective maintenance status. This will doubtless provoke some screaming, and need some wider discussion on gmx-developers, but I suggest

- tools should do one task
- that task must be unit tested
- tools must use the new analysis framework
- if the tool can't use the new analysis framework because the framework does not support enough functionality yet, then we can consider extending it (but the result doesn't have to be as sexy C++ as Teemu writes)
- thus, the tools that get ported will be the ones people want to use enough to put effort into maintaining - but because they only do one thing, and have a test, that maintenance will be easy-or-zero
- users will complain that it's harder to do their work, but that friction generates light and code, as well as heat

Making those conversions viable needs

- a trajectory-reading framework that implements something like the current functionality, behind a sane C++ API #1193
- a representation of rvec that can be used in C++ containers #1017 (various discussions elsewhere, but I suggest we pick something plausible for use in the analysis part of tools and just use it - Eigen looked good last time I considered the question. This will require things like the trajectory-reading code to read into rvec and then convert it, until someone cares enough to fix the situation, but that conversion is unlikely to bound the cost of the analysis. mdrun has separate needs and should be treated separately.)
Just renaming gmxana to contrib does not have any effect. However, not compiling tools by default will mean we quickly lose users, since many users like gromacs for the analysis tools (as well as mdrun performance obviously). Unfortunately there are very few users that contribute code, right now. Most power users work towards their own scientific goals, including myself to a large extent. Looking in trajectoryanalysis/modules all but one of the tools have been coded by Teemu.

Unfortunately porting code to the new framework is a larger barrier than what one might think, it is pretty close to rewriting everything from scratch. Maybe we need to team up with some of our biggest users and get them to contribute more?

By the way, the trajectory analysis framework already hides the actual trajectory reading from the tools, so this is taken care of - if the underlying code changes the tools are not affected. I'm implementing something similar for energy files with guidance from Teemu.

Is anyone still using .g96 format? If it's obsolete lets remove it. I'll ask, just in case, on the mailing list before doing that.

It is not entirely obsolete - I've used it recently as an intermediate when converting LAMMPS trajectories to gromacs, the reason being that g96 has higher precision than both pdb and gro and therefore it is easier to reproduce results from other packages. And pdb does not have velocities of course.

I have used g96 on occasion, and some users certainly do. I don't recall if -f .gro -ndec <whatever> is just as good, though. Given that the implementation for .g96 is of similar quality to that of .gro, I'm happy to keep it. I was against .g87 and .xyz mostly on grounds of implementation quality.

OK, lets keep it then.

Another format question - how about XMGR? Hasn't it been deprecated for a long time in favor of XMGRACE? This is related to #783

I don't think xmgr is used much, but I don't think it's a problem either. Fixing #783 seems unrelated to removing xmgr.

# Related to Bug #860: Checkpoint not created upon reaching time given in maxh added

# Related to Task #1411: Future of thread_mpi added

We should remove tpa, tpb and trj file formats. They've probably been unused for quite a few years, and it would surprise me if they worked correctly.

QM/MM support likely only works with group scheme, so it is also likely to disappear unless someone puts in some work.

More controversially, after thinking more about a comment from Erik the other day, I think we should dump thread-MPI sooner rather than later. There's multiple existing fallbacks for people needing to run on single nodes, and it does add complexity we don't absolutely need.

Gerrit received a related patchset '1' for Issue #1500.

Gerrit received a related patchset '1' for Issue #1500.

- Target version deleted (5.x)

Gerrit received a related patchset '1' for Issue #1500.