GROMACS - Feature #2082

GMX_USE_PLUGINS=ON is incompatible with GMX_PREFER_STATIC_LIBS=ON -- cmake should fail but does not

11/23/2016 11:23 PM - Chris Neale

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Low</td>
</tr>
<tr>
<td>Assignee:</td>
<td>Mark Abraham</td>
</tr>
<tr>
<td>Category:</td>
<td>build system</td>
</tr>
<tr>
<td>Target version:</td>
<td>2016.2</td>
</tr>
<tr>
<td>Difficulty:</td>
<td>simple</td>
</tr>
</tbody>
</table>

**Description**

It appears that when compiling gromacs 5.1.2 with the cmake option `-DGMX_USE_PLUGINS=ON` that also adding the cmake option `-DGMX_PREFER_STATIC_LIBS=ON` causes the plugins to be silently abandoned. Suggest to force failure with a message that `GMX_USE_PLUGINS=ON` is incompatible with `GMX_PREFER_STATIC_LIBS=ON`.

More information is here:
https://mailman-1.sys.kth.se/pipermail/gromacs.org_gmx-users/2016-November/109605.html

**Related issues:**

Related to GROMACS - Bug #598: dlopen - cmake and dependency issues Closed 10/20/2010

**Associated revisions**

Revision f6eacdd8 - 12/20/2016 09:28 PM - Mark Abraham

Improve build system for plugin support

The mdrun-only and prefer-static-libs builds set the default for BUILD_SHARED_LIBS to off, which silently disabled plugin support.

Converted GMX_LOAD_PLUGINS to tri-state ON/OFF/AUTO so that if the preconditions for support are not met we can have suitable behaviour in each case. We now write to the status line only when something relevant changes, and issue a usefully descriptive fatal error only when the user's request cannot be satisfied.

Renamed and reimplemented VMD_QUIETLY as EMIT_STATUS_MESSAGES.

Moved the reorganized code to its own source file.

Refs #2082

Change-Id: l0ad2d8423abbc8d8cb409e74325f2b00831644ea

**History**

#1 - 11/24/2016 01:16 PM - Mark Abraham

That would be a good work-around for users.

I don't know enough about how the machinery for dynamic loading works, but I speculate that a statically linked gmx should still be able to load a dynamic object. If so, the coupling we have between GMX_PREFER_STATIC_LIBS=on setting BUILD_SHARED_LIBS=off which disables plugin support can be reconsidered.

#2 - 11/24/2016 01:49 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2082.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: l0ad2d8423abbc8d8cb409e74325f2b00831644ea
Gerrit URL: https://gerrit.gromacs.org/#/c/6346

#3 - 11/24/2016 01:52 PM - Mark Abraham

GMX_PREFER_STATIC_LIBS merely sets the default for BUILD_SHARED_LIBS to off, but the code seems like it should work with the combination of cmake -DGMX_PREFER_STATIC_LIBS=on -DBUILD_SHARED_LIBS=on -DGMX_LOAD_PLUGINS=on which is one way a user might respond to my new fatal errors. Chris, can you try that please? Should work on any version of GROMACS.
The above patch improves the cmake interface, but we should leave the issue open because the underlying issue of whether we need shared libraries for dynamic loading is not yet considered.

#5 - 11/24/2016 08:50 PM - Chris Neale

Thank you Mark.

Your solution looks like it starts to work -- since cmake gives the message: Using dynamic plugins (e.g. VMD-supported file formats) but then it fails during linking (details below).

-- Checking for dlopen
-- Performing Test HAVE_DLOPEN
-- Performing Test HAVE_DLOPEN - Success
-- Checking for dlopen - found
-- Using dynamic plugins (e.g VMD-supported file formats)
-- Using default binary suffix: ""
-- Using default library suffix: ""

[... snip ...]

[100%] Building C object src/gromacs/CMakeFiles/libgromacs.dir/__/external/thread_mpi/src/once.c.o
[100%] Building C object src/gromacs/CMakeFiles/libgromacs.dir/utility/baseversion-gen.c.o
[100%] Linking CXX shared library ../lib/libgromacs.so
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/libz.a(compress.o): relocation R_X86_64_32 against `.rodata.str1.1' can not be used when making a shared object; recompile with -fPIC
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/libz.a: could not read symbols: Bad value
collect2: ld returned 1 exit status
make[2]: *** [lib/libgromacs.so.1.2.0] Error 1
make[1]: *** [src/gromacs/CMakeFiles/libgromacs.dir/all] Error 2
make: *** [all] Error 2
[ 0%] Built target view_objlib
[ 1%] Built target mdrun_objlib
[ 1%] Linking CXX shared library ../lib/libgromacs.so
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/libz.a(compress.o): relocation R_X86_64_32 against `.rodata.str1.1' can not be used when making a shared object; recompile with -fPIC
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/libz.a: could not read symbols: Bad value
collect2: ld returned 1 exit status
make[2]: *** [lib/libgromacs.so.1.2.0] Error 1
make[1]: *** [src/gromacs/CMakeFiles/libgromacs.dir/all] Error 2
make: *** [all] Error 2

#6 - 11/25/2016 10:27 AM - Roland Schulz
dlopen with a fully static binary might be a problem. But tying it to BUILD_SHARED_LIBS or GMX_PREFER_STATIC_LIBS doesn't seem to make any sense. At most it could be tied to GMX_BUILD_SHARED_EXE but it would be better to simply test whether linking again dl actually works. Note this was introduced by https://gerrit.gromacs.org/#/c/394/

#7 - 11/28/2016 03:19 PM - Mark Abraham

- Related to Bug #598: dlopen - cmake and dependency issues added

#8 - 11/28/2016 04:08 PM - Mark Abraham

@Chris Thanks, that clarifies that the combination of GMX_PREFER_STATIC_LIBS=on and BUILD_SHARED_LIBS=on isn't reliable. (Failed similarly in my hands, also.) Our CMake already warns that linking might fail in the way that it does, but perhaps we should upgrade that to a fatal error.

@Roland Agree things here are a mess, but my googling suggests that dynamic loading can't work without a shared linker, so the simplest thing to do is what my patch does - require BUILD_SHARED_LIBS=on and HAVE_DLOPEN=true. There's something to be said for avoiding position-independent code (e.g. https://cseweb.ucsd.edu/~gbournou/CSE131/the_inside_story_on_shared_libraries_and_dynamic_loading.pdf), but the problem here is that it is unclear whether the implementation of GMX_PREFER_STATIC_LIBS does more harm than good. The plugin support is improved by my patch, and can be improved further, but isn't actually curing Chris's problems.

I propose

- making GMX_LOAD_PLUGINS tri-value (ON/OFF/AUTO) defaulting to AUTO
- that choosing AUTO chooses behaviour consistent with ON or OFF based on the -prerequisite WIN32 or (BUILD_SHARED_LIBS AND HAVE_DLOPEN) (or we could also add a linking test if someone wants to implement that)
- that choosing ON issues a fatal error if the prerequisite is not satisfied (and probably an extra hint if GMX_PREFER_STATIC_LIBS=ON)
that we document plugin support in the install guide

This means that a user who doesn't care about plugin support can't have mysterious problems arising from the value of BUILD_SHARED_LIBS (as pointed out by Teemu at https://gerrit.gromacs.org/#/c/6346/3), or vice versa.

This does not address that a user could simply do cmake -DGMX_PREFER_STATIC_LIBS=ON and now plugins are disabled automatically because BUILD_SHARED_LIBS now defaults to OFF (which should probably be considered for a separate fix). We could have a status message that plugin support is disabled, but it'll mostly not be seen. We shouldn't have a fatal error. If a user like Chris finds out that there's no plugin support, then if they then set cmake -DGMX_LOAD_PLUGINS=ON they'll get the above fatal error + hint, and some docs will help, too.

#9 - 11/28/2016 05:19 PM - Teemu Murtola

I don't mind much about what is the default behavior, but I agree with Roland that it makes little sense to make plugins+static libraries a hard error. Yes, you need the dynamic linker libraries to support dlopen(), but there is nothing that inherently prevents you from linking those libraries themselves statically (or dynamically into a statically linked executable). We are not using plugins that would call back to our through some bindings that would require our own symbols to be found at runtime.

I'm fine with making plugins off by default with static linking (because some compilers can produce mysterious warnings for some combinations), but if the user wants to, they should be able to try make that work. And any warning/fatal error should suggest that turning off the plugin support is also an option, instead of just saying to turn on shared linking.

Making the option tristate can very well help implementing reasonable behavior here.

#10 - 11/28/2016 05:45 PM - Mark Abraham

Teemu Murtola wrote:

I don't mind much about what is the default behavior, but I agree with Roland that it makes little sense to make plugins+static libraries a hard error. Yes, you need the dynamic linker libraries to support dlopen(), but there is nothing that inherently prevents you from linking those libraries themselves statically (or dynamically into a statically linked executable). We are not using plugins that would call back to our through some bindings that would require our own symbols to be found at runtime.

Agree that it sounds plausible to engineer, but it's a corner case. People who want static linking are generally running mdrun for maximum performance, and people who want the VMD plugins are generally running tools for setup and analysis. I don't think it's a good investment of developer time for us to cater for both needs in the same build.

I'm fine with making plugins off by default with static linking (because some compilers can produce mysterious warnings for some combinations), but if the user wants to, they should be able to try make that work.

Permitting such an attempt means it must be possible for a combination of CMake options to lead to linking failure (ideally of dlopen() tested at configure time). It's not clear to me that we can do that while also avoiding some other user being confused by such failure. I think requiring dynamic linking for plugin support on *nix is a reasonable compromise. Someone keen enough to try to make a statically-linked binary work with dynamic loading can probably hack the CMake, too...

And any warning/fatal error should suggest that turning off the plugin support is also an option, instead of just saying to turn on shared linking.

Sure

#11 - 11/28/2016 08:17 PM - Teemu Murtola

Mark Abraham wrote:

Agree that it sounds plausible to engineer, but it's a corner case. People who want static linking are generally running mdrun for maximum performance, and people who want the VMD plugins are generally running tools for setup and analysis. I don't think it's a good investment of developer time for us to cater for both needs in the same build.

Why is it better use of developer time to explicitly write code to check for and prevent such corner cases, and deal with problems this explicit check+prevent causes in combination with other things, rather than just allow such a corner case to exist if the user explicitly requests it? All me or Roland is saying that the fatal error that can produce false negatives should be removed. That is less work than this discussion...

Permitting such an attempt means it must be possible for a combination of CMake options to lead to linking failure (ideally of dlopen() tested at configure time). It's not clear to me that we can do that while also avoiding some other user being confused by such failure. I think requiring dynamic linking for plugin support on *nix is a reasonable compromise. Someone keen enough to try to make a statically-linked binary work with dynamic loading can probably hack the CMake, too...

I don't understand your point. All kinds of misconfigurations can lead to all kinds of failures. If the user is only getting such a failure when they explicitly request an unusual combination, they amount of people getting confused should be minimal.

There is an explicit use case from Chris that he seems to want to combine plugins with static linking. Now, instead of trying to honor that explicit request (which could very well work; I did test and compiling static gmx with dlopen() is perfectly fine on my OS X), your fatal error suggest turning on...
shared libraries, which leads to completely unrelated problems about mismatching libraries for that combination.

#12 - 11/28/2016 08:46 PM - Chris Neale

Just to clarify my reason for posting:

I have no need for static libs with plugins. I'm just pointing out that while I probably build gromacs more than the average user, I still had to turn to the gmx users list to figure out how to compile with the plugins. It's obviously my mistake, and to some it might seem obvious that shared libs are required to link the plugins (it's obvious to me now but was not before). In any event, the mailing list solution worked since I got help from Mark (and now that solution is archived and searchable).

#13 - 11/28/2016 10:14 PM - Mark Abraham

@Teemu Yes, I already agreed that my initial proposal introduces an undesirable way to get a new fatal error. It would be better to focus on my proposal in comment 8.

Specifically, I am not going to propose that we remove any of the couplings of GMX_PREFER_STATIC_LIBS and GMX_LOAD_PLUGINS with BUILD_SHARED_LIBS because I don't have time to test those well, regardless of which branch they might go in. Requiring BUILD_SHARED_LIBS=ON for plugin support hits the important targets easily.

#14 - 11/29/2016 07:27 AM - Teemu Murtola

All the discussion here is relevant in the context of your new proposal in comment 8, in particular the third bullet. All other aspects of your proposal seem fine. If you insist on keeping that fatal error for cases where we know it can produce false negatives, it should at least be separated from the case where something concrete has failed. And the message should tell the user that it's an arbitrary limitation in the build system, not something that doesn't work.

With your reasoning, we should make it a fatal error to even attempt to link with any static library with BUILD_SHARED_LIBS=on, because we don't have complete test coverage of every possible combination in CMake. Except that for this case, we have concrete evidence (Chris's error) that this can fail, but for the dlopen() case, we have only vague suspicions (and a concrete case where it does work).

#15 - 12/09/2016 02:24 AM - Gerrit Code Review Bot

Gerrit received a related patchset '6' for Issue #2082
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~release-2016~I0ad2d8423abbc8d8cb409e74325f2b00831644ea
Gerrit URL: https://gerrit.gromacs.org/6346

#16 - 01/20/2017 02:03 PM - Mark Abraham

- Status changed from Fix uploaded to Closed

Fix is now merged