GROMACS - Feature #911

implement CMake option to enable fully static binaries

04/03/2012 03:22 PM - Szilárd Páll

Status:	Closed
Priority:	Normal
Assignee:	Szilárd Páll
Category:	build system
Target version:	5.1
Difficulty:	uncategorized

Description

On some systems like Cray statically linked binaries are either a must or highly recommended.

Below are the steps required to get static linking to work; the method is not foolproof, it can fail in some cases (e.g. if some external libraries get shared version detected). We should attempt to automate this as much as possible.

- set the target properties LINK_SEARCH_START_STATIC and LINK_SEARCH_END_STATIC, e.g:

  set_target_properties(mdrun PROPERTIES LINK_SEARCH_START_STATIC ON)
  set_target_properties(mdrun PROPERTIES LINK_SEARCH_END_STATIC ON)

- configure with -static and disable CMake RPATH:

  cmake /path/to/gromacs/source -DCMAKE_PREFIX_PATH=/path/to/fftw -DGMX_PREFER_STATIC_LIBS=ON -DCMAKE_C_FLAGS="-static" -DCMAKE_CXX_FLAGS="-static" -DCMAKE_SKIP_RPATH=YES

Related issues:

Related to GROMACS - Feature #1641: Add toolchain file for Cray systems	**New**	**11/11/2014**

Associated revisions

Revision 88fda475 - 05/13/2015 10:45 AM - Roland Schulz

Facilitate linking of static binaries

Minimal solution. The user has to manually set both
-DBUILD_SHARED_EXE=no and CFLAGS=CXXFLAGS=-static, perhaps manage
their own toolchain, and certainly make static libraries available for
all dependencies. Also does not auto-detect if compiler defaults to
static (Cray). Works better than LINK_SEARCH_END_STATIC because
otherwise dynamic flags can be added to the middle if some libraries
in default search path exist as both dyanmic and shared.

Fixes #911
Related to #1641

Change-Id: If7b8192b44c33c861f126e3422df04388d2f2be5

History

#1 - 04/03/2012 03:22 PM - Szilárd Páll

- Assignee deleted (Rossen Apostolov)

#2 - 08/04/2012 08:33 PM - Roland Schulz

For GMX_PREFER_STATIC_LIBS we use

```
SET(CMAKE_FIND_LIBRARY_SUFFIXES .a ${CMAKE_FIND_LIBRARY_SUFFIXES})
```

Any reason not to use

```
SET(CMAKE_FIND_LIBRARY_SUFFIXES .a)
```

for fully static libraries?

LINK_SEARCH_END_STATIC is only available starting with cmake 2.8.5. In what cases are LINK_SEARCH_START_STATIC/LINK_SEARCH_END_STATIC needed? For me it always worked without. I thought that giving "-static" automatically makes it work.

While the rpath is not necessary for static binaries I don't think it hurts. Again it just works for me.

#3 - 10/08/2012 03:07 PM - Szilárd Páll

Roland Schulz wrote:

> For GMX_PREFER_STATIC_LIBS we use
> [...]
> Any reason not to use
> [...]
> for fully static libraries?

Hmm, I didn't want to just override the content of CMAKE_FIND_LIBRARY_SUFFIXES. Would that be always safe?

> LINK_SEARCH_END_STATIC is only available starting with cmake 2.8.5. In what cases are LINK_SEARCH_START_STATIC/LINK_SEARCH_END_STATIC needed? For me it always worked without. I thought that giving "-static" automatically makes it work.

Those are needed to link system libraries statically:
http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_START_STATIC
http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_END_STATIC

Now, what I'm not sure about anymore whether both are always needed or it's better to set both because in some cases the former, in other cases the latter is needed.

> While the rpath is not necessary for static binaries I don't think it hurts. Again it just works for me.

Not sure about the exact source of the idea of disabling RPATH, but as far as I remember it was recommended on the CMake mailing list.

#4 - 10/08/2012 11:57 PM - Roland Schulz

Szilárd Páll wrote:

> Roland Schulz wrote:
>
>> For GMX_PREFER_STATIC_LIBS we use
>> [...]
>> Any reason not to use
>> [...]
>> for fully static libraries?
>
> Hmm, I didn't want to just override the content of CMAKE_FIND_LIBRARY_SUFFIXES. Would that be always safe?

Not sure. But it is used by others successful: http://www.mail-archive.com/cmake@cmake.org/msg21354.html

>> LINK_SEARCH_END_STATIC is only available starting with cmake 2.8.5. In what cases are LINK_SEARCH_START_STATIC/LINK_SEARCH_END_STATIC needed? For me it always worked without. I thought that giving "-static" automatically makes it work.
>
> Those are needed to link system libraries statically:
> http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_START_STATIC
> http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_END_STATIC

That means it is not a proper solution if we want it to work with 2.8.0. Also there are other problems:
http://www.cmake.org/pipermail/cmake/2010-November/040755.html

>> While the rpath is not necessary for static binaries I don't think it hurts. Again it just works for me.
>
> Not sure about the exact source of the idea of disabling RPATH, but as far as I remember it was recommended on the CMake mailing list.

I guess it doesn't hurt since it is obviously not needed.

#5 - 10/09/2012 12:11 AM - Szilárd Páll

Roland Schulz wrote:

> Szilárd Páll wrote:
>
> > Roland Schulz wrote:
> >
> > > For GMX_PREFER_STATIC_LIBS we use
> > > [...]
> > > Any reason not to use
> > > [...]
> > > for fully static libraries?
> >
> > Hmm, I didn't want to just override the content of CMAKE_FIND_LIBRARY_SUFFIXES. Would that be always safe?
>
> Not sure. But it is used by others successful: http://www.mail-archive.com/cmake@cmake.org/msg21354.html
>
> > > LINK_SEARCH_END_STATIC is only available starting with cmake 2.8.5. In what cases are LINK_SEARCH_START_STATIC/LINK_SEARCH_END_STATIC needed? For me it always worked without. I thought that giving "-static" automatically makes it work.
> >
> > Those are needed to link system libraries statically:
> > http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_START_STATIC
> > http://www.cmake.org/cmake/help/v2.8.9/cmake.html#prop_tgt:LINK_SEARCH_END_STATIC
>
> That means it is not a proper solution if we want it to work with 2.8.0. Also there are other problems:
> http://www.cmake.org/pipermail/cmake/2010-November/040755.html

Yeah, that looks familiar, it's my thread after all :) I have not seen the -lgcc_s errors for a year or so.

Anyway, my point was not to implement a rock-solid feature because that's simply not possible (the build will fail if not all external dependencies are detected as static archives), but to make a fair attempt to build a fully static mdrun (perhaps tools as well) and control it through an advanced variable.

And my question was whether this is useful or not. For me and a handful of other working on quirky Crays it would be useful to not have to patch CMake sources before compiling static binaries.

> > > While the rpath is not necessary for static binaries I don't think it hurts. Again it just works for me.
> >
> > Not sure about the exact source of the idea of disabling RPATH, but as far as I remember it was recommended on the CMake mailing list.
>
> I guess it doesn't hurt since it is obviously not needed.

#6 - 10/09/2012 12:25 AM - Roland Schulz

> Yeah, that looks familiar, it's my thread after all :) I have not seen the -lgcc_s errors for a year or so.

Didn't see that ;-)

> And my question was whether this is useful or not. For me and a handful of other working on quirky Crays it would be useful to not have to patch CMake sources before compiling static binaries.

I'm surprised you need to do anything special on Cray. All the Cray I have access to automatically add the "-static" in the "cc" compiler wrapper and it just works for me.

#7 - 10/09/2012 12:30 AM - Szilárd Páll

> > And my question was whether this is useful or not. For me and a handful of other working on quirky Crays it would be useful to not have to patch CMake sources before compiling static binaries.

> I'm surprised you need to do anything special on Cray. All the Cray I have access to automatically add the "-static" in the "cc" compiler wrapper and it just works for me.

Newer Crays don't, in particular on an XK6 (where it would even be impossible due to CUDA not being distributed statically) and both XE6 machines I have access to. I guess I'll have to ask the question differently: do you have anything against such a feature?

#8 - 10/09/2012 12:34 AM - Roland Schulz

No. Since it is anyhow only for a small group of people it might be OK if it only works with >=2.8.5 or it doesn't always work. As long as it doesn't have any effect when it is disabled.

#9 - 11/05/2012 10:40 PM - Roland Schulz

Somehow they changed something on Hopper and now you need the LINK_SEARCH_END_STATIC there too. And it works by itself without problems. So I think it is a good idea to have a option which enables it. I don't think it should be enabled by GMX_PREFER_STATIC_LIBS (I don't think you suggested that - instead I thought originally this would be a good idea). Instead we need a separate flag for this (e.g. GMX_FULLY_STATIC). This should also add "-static". This later isn't needed on Cray because it already gets added by their compiler wrapper but is useful in general. E.g. it avoid the gcc_s problem on non Cray machines. We should print a warning on cmake <2.8.5 that the option GMX_FULLY_STATIC isn't working.

#10 - 11/08/2012 03:07 PM - Roland Schulz

I looked it up incorrectly 3 months ago for comment #2. LINK_SEARCH_END_STATIC is available in 2.8.0 - only LINK_SEARCH_START_STATIC was added in 2.8.5 (http://www.cmake.org/Wiki/CMake_Version_Compatibility_Matrix/Properties#Properties_on_Targets). But I don't think we actually need LINK_SEARCH_START_STATIC. So fixing this issue should be as simple as adding a new option which activates LINK_SEARCH_START_STATIC and adds "-static".

#11 - 11/08/2012 03:18 PM - Szilárd Páll

Roland Schulz wrote:

> I looked it up incorrectly 3 months ago for comment #2. LINK_SEARCH_END_STATIC is available in 2.8.0 - only LINK_SEARCH_START_STATIC was added in 2.8.5 (http://www.cmake.org/Wiki/CMake_Version_Compatibility_Matrix/Properties#Properties_on_Targets). But I don't think we actually need LINK_SEARCH_START_STATIC. So fixing this issue should be as simple as adding a new option which activates LINK_SEARCH_START_STATIC and adds "-static".

I was just looking at that earlier today. What I don't understand is how does the -Bstatic affect the linking when at the beginning or end of the library list?

#12 - 11/08/2012 03:31 PM - Roland Schulz

What I assume (inferred from the behavior without really knowing how it works) is that the libraries added by the linker by default (e.g. libgcc,glibc,...) are added to the end, and are affected by the last Bdynamic/Bstatic in the list. Thus without the LINK_SEARCH_END_STATIC the default libs are linked dynamic and with it they are linked static. I don't think LINK_SEARCH_START_STATIC has any effect for any of the compilers we use. The only place it seems to matter is for odd custom tool-chains: http://cmake.3232098.n2.nabble.com/To-avoid-target-link-libraries-magic-td6192280.html . But I don't think it hurts either (neither with version which support it nor with those which don't). So I would simply set it and don't worry about that it isn't supported with older cmake because 99% of users probably don't need it anyhow.

#13 - 11/08/2012 03:36 PM - Szilárd Páll

I'm starting to get it now. The LINK_SEARCH_END_STATIC affects **system** libs while the LINK_SEARCH_START_ affects the -lXXX library list. Assuming that this is the case, we should use both (if supported).

#14 - 11/08/2012 03:46 PM - Roland Schulz

Well if cmake finds static libraries with PREFER_STATIC then it already adds the Bstatic before those libraries (and if it doesn't find the static library something is anyhow wrong). That's why I think it isn't necessary to have LINK_SEARCH_START_STATIC. But as I wrote it shouldn't hurt and might be helpful in some odd cases.

#15 - 01/07/2013 03:19 PM - Mark Abraham

Are we able to get something done here for 4.6?

#16 - 01/08/2013 06:12 AM - Roland Schulz

I think we agreed on:
- add a new option GMX_STATIC_BINARY (probably better name than GMX_FULLY_STATIC)
- add -static (if supported compiler option), LINK_SEARCH_END_STATIC and LINK_SEARCH_START_STATIC (if cmake>=2.8.5)

So should be very easy. Who wants to do it?

#17 - 01/08/2013 02:50 PM - Szilárd Páll

Roland Schulz wrote:

> I think we agreed on:
> - add a new option GMX_STATIC_BINARY (probably better name than GMX_FULLY_STATIC)
> - add -static (if supported compiler option), LINK_SEARCH_END_STATIC and LINK_SEARCH_START_STATIC (if cmake>=2.8.5)

Yes, that is the plan.

> So should be very easy. Who wants to do it?

I wanted to do it, but have been quite busy with other stuff and forgot about it. Will do it asap.

#18 - 01/08/2013 02:50 PM - Szilárd Páll

- Status changed from New to In Progress

- Assignee set to Szilárd Páll

#19 - 01/18/2013 04:07 PM - Erik Lindahl

- Target version changed from 4.6 to future

#20 - 02/04/2013 12:09 AM - Roland Schulz

The reason why LINK_SEARCH_END_STATIC sometimes is required and sometimes not, seems to be related to whether any library in standard locations is used. In that case cmake uses "-Wl,-Bstatic -lsomelib" instead of "/usr/lib/libsomelib.a". Now that xml is removed and fftw isn't in the default location on Cray, it seems to be no problem on Cray.

#21 - 05/22/2013 05:56 AM - Mark Abraham

- Target version changed from future to 4.6.x

#22 - 11/07/2013 09:34 PM - Mark Abraham

- Target version changed from 4.6.x to future

I seem to be able to build on Cray without problems, but my libraries were standard or own-fftw.

#23 - 11/11/2013 09:03 PM - Szilárd Páll

Mark Abraham wrote:

> I seem to be able to build on Cray without problems, but my libraries were standard or own-fftw.

That's because on Crays the compiler wrappers force static builds whenever they can without being asked to do so, but this of course only works if all external libraries in their static form. However, even on Crays without turning off RPATH you won't be able to do make install because cmake thinks that it built dynamically linked binaries.

Note that you may not even be using the "own" FFTW because the Cray toolchain will link binaries against the scientific libs which contain Cray's own FFTW (which is anyway tuned and typically faster than the standard FFTW). Hence, I guess it's a matter or linking order whether you get to use your or Cray's FFTW.

#24 - 11/11/2013 09:57 PM - Mark Abraham

So we should expect/design the CMake detection to pick up that the toolchain provides FFTW even if there's nothing specific in any of the the paths? Does anybody know this happens?

#25 - 11/11/2013 10:20 PM - Roland Schulz

Currently it isn't checked. But it could be easily added by testing with CHECK_FUNCTION_EXISTS before looking for the library. But I think it isn't strictly related to the issue of having a static binary.

#26 - 11/12/2013 02:33 PM - Szilárd Páll

Mark Abraham wrote:

> So we should expect/design the CMake detection to pick up that the toolchain provides FFTW even if there's nothing specific in any of the the paths? Does anybody know this happens?

Given that it will be a rare thing that compiler toolchains silently pull in loads of libraries "thought to be good for you" (does it happen on other HPC machine besides Cray?), unless it's trivial I would considere such an issue (if filed) a low priority one.

Note that I **know** this happens (at least on the CSCS XE6 Rosa and XK7 Toedi) because a while ago I wanted to benchmark standard FFTW against Cray's modified FFTW and after some struggling I gave up because I could not get the linking order tweaked appropriately.

...and indeed, this aspect is not really related to the current issue.

#27 - 11/13/2013 01:36 PM - Rossen Apostolov

Regarding the linking order - if you need to add a library right at the end of the link line, at least on some systems (e.g. SuperMUC) CMAKE_EXE_LINKER_FLAGS doesn't help. Instead one needs to use CMAKE_C_STANDARD_LIBRARIES to pass the arguments.

#28 - 06/12/2014 04:08 PM - Rossen Apostolov

Update - in 5.0.x and master one needs to use CMAKE_C_STANDARD_LIBRARIES_INIT instead.

#29 - 06/13/2014 12:12 PM - Szilárd Páll

- Status changed from In Progress to Accepted

#30 - 10/29/2014 10:21 AM - Mark Abraham

A somewhat related issue is that on Cray systems, the early tests by CMake for a functional compiler will pick up the CMake-standard Modules/Platform/Linux-xyz-lang.cmake files, which fails completely when Intel is the base compiler for the Cray wrapper, because these files hardcode the use of CMAKE_SHARED_LIBRARY_LINK_CXX_FLAGS=-rdynamic. Fix for 5.0 in progress

#31 - 10/29/2014 11:29 AM - Gerrit Code Review Bot

Gerrit received a related DRAFT patchset '1' for Issue #911.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: Id5e9f6594bb481617e58d1ed4e79fdc692689ece
Gerrit URL: https://gerrit.gromacs.org/4188

#32 - 11/18/2014 04:26 PM - Mark Abraham

- Related to Feature #1641: Add toolchain file for Cray systems added

#33 - 11/18/2014 05:55 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #911.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: I47b4524e9c33f52f20b0f54082d290ee22d9993b
Gerrit URL: https://gerrit.gromacs.org/4227

#34 - 11/26/2014 07:27 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #911.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: I47b4524e9c33f52f20b0f54082d290ee22d9993b
Gerrit URL: https://gerrit.gromacs.org/4242

#35 - 12/12/2014 04:19 PM - Mark Abraham

- Description updated

#36 - 03/17/2015 05:12 AM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #911.
Uploader: Roland Schulz (roland@rschulz.eu)
Change-Id: If7b8192b44c33c861f126e3422df04388d2f2be5
Gerrit URL: https://gerrit.gromacs.org/4496

#37 - 06/20/2015 11:48 PM - Szilárd Páll

- Status changed from Accepted to Closed

- Target version changed from future to 5.1