GROMACS - Bug #1067

ICC links against FFTW-MKL wrapper instead of FFTW library

12/11/2012 05:09 AM - Roland Schulz

Status: Closed
Priority: Normal
Assignee: Mark Abraham
Category: build system
Target version: 4.6.1
Affected version:

Description

Compiling with ICC 13 links by default against MKL (for BLAS and LAPACK). With GMX_FFT_LIBRARY this causes mdrun to use the FFTW-MKL wrapper instead of the FFTW library cmake found. This can be seen by the version output:

FFT library: FFTW 3.2 wrappers to MKL

I tested this with BUILD_SHARED_LIBS=no but might very well be also with shared libraries. The FFTW-MKL wrapper is slower than MKL which is slower than FFTW. So this is a bad default. Also it is confusing because the cmake output and the CMakeCache both have the real fftw library.

I'm not sure whether it is possible to guarantee the linking order to make sure that the real FFTW and not the wrapper is used. If not we might want to disallow MKL BLAS/LAPACK with FFTW or at least print a warning.

Related issues:

Related to GROMACS - Bug #1186: install guide should mention how to configure... Closed 03/11/2013

Associated revisions

Revision 8aba41b6 - 03/05/2013 06:57 PM - Roland Schulz

Fixes linkage with FFTW + MKL for BLAS/LAPACK

The order of linking was incorrect, when using both mkl and fftw causing the mkl fftw wrapper to be used instead of fftw itself.

Removes all unnecessary transitive libraries (cmake adds those automatically). Simplifies the target_link_libraries lines, and makes it much easier to change the fft/blas link order.

Only remaining transitive libraries are gmx/md for binaries. This is required because OpenMP_LINKER_FLAGS needs to be at the end but transitive libraries are added at the end. Thus without listing gmx/md manually the order is incorrect.

Also fixes that nbnxn_cuda has cuda_tools listed as library dependency. Was prior working without it but now is needed and is also anyhow the more correct way to do it.

Fixes #1067

Change-Id: I639b16f1460d27a15eb72cc00674c041266f6624

History

#1 - 12/12/2012 02:23 PM - Mark Abraham
- Assignee set to Mark Abraham
- Target version set to 4.6

I'm doing an overhaul of FFT setup (and boy do I know a lot more about how CMake handles linking now), since version 11 and up of the Intel compilers support "-mkl" to do all the required magic. I already intended that to expand into cleaning up this area. Now it's official! :-) I will have to get icc 13 to check that out.

#2 - 01/09/2013 03:16 AM - Szilárd Páll

It does happen with shared libs and Intel Compilers + MKL 12.1 as well. Turning off external BLAS and LAPACK fixes it. As an intermediate fix i'd
suggest turning off external BLAS/LAPACK (which is anyway not crucial) with icc + fftw.

#3 - 01/18/2013 03:35 PM - Erik Lindahl
- Target version changed from 4.6 to 4.6.1

#4 - 03/04/2013 05:22 PM - Mark Abraham
Roland Schulz wrote:

  Compiling with ICC 13 links by default against MKL (for BLAS and LAPACK).

Do you mean that our default behaviour for external BLAS+LAPACK finds stuff provided by icc 13, even if the command line is not using -mkl?

  With GMX_FFT_LIBRARY this causes mdrun to use the FFTW-MKL wrapper instead of the FFTW library cmake found. This can be seen by the version output:

  FFT library:        FFTW 3.2 wrappers to MKL
  I tested this with BUILD_SHARED_LIBS=no but might very well be also with shared libraries. The FFTW-MKL wrapper is slower than MKL which is slower than FFTW. So this is a bad default. Also it is confusing because the cmake output and the CMakeCache both have the real fftw library.

  At least testing for that string gives us a way to detect the problem. I still don't know how we can use MKL in icc 13 for BLAS+LAPACK, together with real FFTW.

  I'm not sure whether it is possible to guarantee the linking order to make sure that the real FFTW and not the wrapper is used. If not we might want to disallow MKL BLAS/LAPACK with FFTW or at least print a warning.

Yeah, not clear.

#5 - 03/04/2013 05:32 PM - Mark Abraham

Part of the problem is that MKL provides two features useful to GROMACS - linear algebra and FFT. Currently, GROMACS detection doesn't deal properly with this. The GROMACS CMake user interface mostly assumes different libraries will provide these features. I think the form of the solution should look like:

  a) Did the user specify linking to MKL? If so, check it will work and use it for linear algebra and FFT
  b) Search for external linear algebra libraries, fall back on internal (or the opposite)
  c) Download and build FFTW if the user said to do that
  d) Search for external FFTW. If found, check it will work (and is not the MKL wrapper) and use it.
  e) Fall back on fftpack (and warn the user)

#6 - 03/04/2013 10:06 PM - Szilárd Páll

Mark Abraham wrote:

  b) Search for external linear algebra libraries, fall back on internal (or the opposite)

The problem with this step is that if MKL was not requested, if the Intel Compiler environment variables are sourced, FindLAPACK and FindBLAS will pick up the MKL libs by default and with this, regardless of what the FFT default or user choice was, the MKL will be used for FFTs through the FFTW wrapper. Unless we can figure out how to prevent this unexpected change from FFTW to MKL to happen, we should disable the use of external BLAS and LAPACK by default - these are anyway used in code which is not that performance-sensitive.

#7 - 03/05/2013 09:09 AM - Roland Schulz
- Status changed from New to In Progress

#8 - 03/05/2013 09:21 AM - Roland Schulz

Possible solution: [https://gerrit.gromacs.org/#/c/2207/1](https://gerrit.gromacs.org/#/c/2207/1)

#9 - 03/08/2013 04:35 PM - Mark Abraham

I did a series of tests of Roland's patch set 3 on icc 13.0.0 20120731. Having correctly sourced compilervars.sh, I observed the following to work correctly (link to system fftw3 and blas+lapack):

  CC=icc CXX=icpc cmake -DMKL_LIBRARIES=$MKLROOT -DGMX_FFT_LIBRARY=fftw3
  CC=icc CXX=icpc cmake -DCMAKE_PREFIX_PATH=$MKLROOT -DGMX_FFT_LIBRARY=fftw3
  CC=icc CXX=icpc cmake -DGMX_FFT_LIBRARY=fftw3

  I wasn't able to succeed in building this patch with icc13 using any of:
CC=icc CXX=icpc cmake .. -DMKL_LIBRARIES=$MKLROOT -DGMX_FFT_LIBRARY=mkl
(MKL was detected, but no blas/lapack was actually available at link time because the link target was a directory, not a library. CMake warned about this; make later failed.)

CC=icc CXX=icpc cmake .. -DMKL_LIBRARIES="$MKLROOT/lib/intel64" -DGMX_FFT_LIBRARY=mkl
(Same again.)

CC=icc CXX=icpc cmake .. -DMKL_LIBRARIES="$MKLROOT/lib/intel64" -DGMX_FFT_LIBRARY=mkl
-DCMAKE_EXE_LINKER_FLAGS="-mkl=sequential"
(MKL detected, CMake gave warnings about linking to directories, not libraries. Linking and running was fine. Mikhail reported in #1110 that it worked, so I tried it again on the parent commit, and again it did build and run properly despite the CMake warnings. So, these warnings should be eliminated (in a new patch, probably also fixing #1110). Probably, FindMKL.cmake is broken, and now redundant?).

When I tried the last configuration again but specifying internal BLAS and LAPACK, I got the MKL versions still. There's a case for this being the correct behaviour.

As expected, with the following command, MKL wasn't detected at all, so configuring failed:
CC=icc CXX=icpc cmake .. -DCMAKE_PREFIX_PATH=$MKLROOT -DGMX_FFT_LIBRARY=mkl

Finally,
CC=icc CXX=icpc cmake .. -DMKL_LIBRARIES="$MKLROOT/lib/intel64" -DGMX_FFT_LIBRARY=fftw3
-DCMAKE_EXE_LINKER_FLAGS="-mkl=sequential"
gave me system fftw3, blas and lapack. So Roland's fix certainly works for picking up the right FFT. In theory, we should be able to get FFTW for FFT and MKL for linear algebra, but apparently that would need further work.

#10 - 03/08/2013 05:25 PM - Szilárd Páll
Note that setting -DMKL_LIBRARIES="$MKLROOT/lib/intel64" can not and should not work. For the explanation see #1110.

#11 - 04/29/2013 08:06 PM - Mark Abraham
- Status changed from In Progress to Resolved
- Affected version set to 4.6

#12 - 06/10/2013 08:14 PM - Mark Abraham
Note, was actually fixed right after 4.6.1.

#13 - 12/16/2013 05:38 PM - Rossen Apostolov
- Status changed from Resolved to Closed