Gromacs - Bug #1673

Task # 1855 (New): Convert preprocessor use so that symbols are always defined

mis-use of simd.h after refactoring

01/18/2015 11:09 PM - Mark Abraham

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</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>mdrun</td>
</tr>
<tr>
<td>Target version:</td>
<td>5.1.1</td>
</tr>
<tr>
<td>Affected version:</td>
<td>5.x</td>
</tr>
</tbody>
</table>

### Affected version - extra info:

- Affected version: 5.x

### Description

Various code needs to include simd.h in order for their #ifdef to work correctly. Our nice fallbacks when SIMD isn't supported work against us if we don't enforce that simd.h is included. The refactoring of the listed-forces module made one such error before 5.1 (fixed at https://gerrit.gromacs.org/#/c/4387/2). The development of the SIMD module made another in forcerec.c that maybe means PME short-ranged kernels run slower than planned on FMA-capable x86 (AMD and Haswell) for Gromacs 5.x to date (but maybe it got simd.h transitively?).

There was a similar bug with some testing code I wrote that required GMX_MPI to work, but didn't #include "config.h" after I split into multiple development files. We now check for correct use of config.h with make check-source, and the same approach will work for SIMD as well. Proposed fix incoming.

### Related issues:

- Related to Gromacs - Bug #1686: bonded force regression

### Associated revisions

**Revision 44cc879c - 01/23/2015 05:59 PM - Mark Abraham**

Change some listed-forces headers to be library-only

Some analysis tools call functions declared here, so they can't be module-only (yet). Clients of libgromacs should only use functions from listed-forces.h.

Moved ftype_is_bonded_potential() accordingly.

This will also make it easier to re-instate SIMD support for angles and dihedrals, because there has to be an intra-module header that declares the SIMD versions of those functions, and it cannot also be an installed header if it depends on simd.h.

Refs #1673

Change-Id: la2815b57f712fbeb1e294dbea3231c6c9bfc32dc

**Revision 3df2b06e - 02/02/2015 02:04 PM - Mark Abraham**

Fix use of simd.h

02/13/2015

11/25/2015
TPI correctness check was inactive.

Use of SIMD preprocessor symbols in forcerec.c was OK because it got simd.h from nbnxn_simd.h, but also fixed the erroneous use of transitive inclusion of simd.h just in case that helps someone.

Fixes #1673

Change-Id: laf42a4ec420139485de3e509662ca0892fa662c5

---

Revision 2755334d - 02/06/2015 01:21 PM - Berk Hess

Re-enabled SIMD for angles and dihedrals

The SIMD acceleration for angles and dihedrals was accidentally disabled recently during code reorganization, since the SIMD header file was not includes and thus the SIMD macro was not set.

Fixes #1673

Change-Id: l04ddde71c9dd1a846ecf088a5c973e9a23846a52

---

Revision 33619c65 - 05/19/2015 04:43 PM - Mark Abraham

Add checks for correct use of SIMD headers.

Generalized the checking in check-source.py to consider "master"-type include files other than config.h. Any source file using a symbol that is correctly defined (or not) only by including such a header now must include it. Conversely, if no such symbol is used, then that header may not be included.

Fixed issues where simd.h, simd_math.h and pme-simd.h files were being acquired transitively. We should not rely on transitive #includes, because they're too easy to break under maintenance. Unfortunately, the use of preprocessor kernel generation prevents these checks working, but we should fix the generation, and not the checker.

Any actual problems from failure to use appropriate SIMD support have been resolved in other commits.

Noted the existence of the new checker in the developer SIMD and check-source docs.

Refs #1673

Change-Id: ld1858ba97f6fle12177c0290e2f65dbabcc41e2f
#2 - 01/19/2015 08:55 AM - Mark Abraham

A further aspect to consider here is that we should be able to use something to verify that we don't test for preprocessing symbols (at least those with a GMX_ prefix) that we do not define anywhere. That would be enough to catch #1603 (and one or two similar pre-release LJPM kernel issues).

It would be better still to check that those definitions happen earlier in the translation unit, but the use of the preprocessor for meta-programming makes implementing that painful.

#4 - 01/19/2015 04:54 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #1673.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: Ia2815b57f712fbeb1e294d8f3231c6c9cfc32dc
Gerrit URL: https://gerrit.gromacs.org/4395

#5 - 01/20/2015 01:25 PM - Mark Abraham

- Subject changed from mis-use of simd.h after refactoring (possible performance loss) to mis-use of simd.h after refactoring

No performance was actually lost - forcerec.c got simd.h from nbnxn_simd.h in release-5-0

#6 - 01/20/2015 05:45 PM - Szilárd Páll

Mark Abraham wrote:

No performance was actually lost - forcerec.c got simd.h from nbnxn_simd.h in release-5-0

Good to hear - I would have been quite surprised if it turned out otherwise as I did some (although not extensive) performance evaluation before/around the 5.0 release and compared against my historical 4.6 performance numbers.
Gerrit received a related patchset '4' for Issue #1673
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: I04ddde71c9dd1a846ecf088a5c973e9a23846a52
Gerrit URL: https://gerrit.gromacs.org/4387

#8 - 01/30/2015 10:46 AM - Teemu Murtola

gcc has -Wundef to warn about using an undefined preprocessor symbol in an #if. With suitably structured #defines and #if instead of #ifdef, this should also be able to catch these issues. Not sure how many bogus warnings it would currently generate, though.

#9 - 02/11/2015 03:23 AM - Szilárd Páll

I still see at least 10% performance difference between the bondeds in 5.0 and current master. Not even sure if the former bonded f now called listed f cycle-count contains the same thing so backward comparability using the subcounters may not even be impossible.

#10 - 02/25/2015 03:00 PM - Mark Abraham

Szilárd Páll wrote:

I still see at least 10% performance difference between the bondeds in 5.0 and current master. Not even sure if the former bonded f now called listed f cycle-count contains the same thing so backward comparability using the subcounters may not even be impossible.

See commit message for 19fb85c001928e6de214218bf3047445f056bb7. Position restraints were added, because they should have been there all along. Nothing else should have changed, now that we have restored all of the SIMD support.

#11 - 02/25/2015 05:56 PM - Szilárd Páll

Mark Abraham wrote:

Szilárd Páll wrote:

I still see at least 10% performance difference between the bondeds in 5.0 and current master. Not even sure if the former bonded f now called listed f cycle-count contains the same thing so backward comparability using the subcounters may not even be impossible.

See commit message for 19fb85c001928e6de214218bf3047445f056bb7. Position restraints were added, because they should have been there all along.

I assume by "there" means in the counter renamed from "Bonded" to "Listed". Frankly the "should have been there" statement is IMHO questionable and it is something that some of us disagree with it (see comments on change #4461).

I was against such that change from the beginning and brought up the very reasons that materialized now: it made comparison unnecessarily hard for the sake of matching the naming of timed entities with aspects of the (new) implementation.

I've asked several times (last #4461) that bonded and other listed interaction counters are kept separate both because (to us) it seems to make more sense and because backward comparability is useful and important. IMO it's not the shared algorithm or module implementation what matters when picking the scope of subcounters and changing the content of counters for no other good reason than to reflect implementation details has obviously created a barrier for cross checking performance against 5.0.

Nothing else should have changed, now that we have restored all of the SIMD support.
The code got reorganized too, did it not? So it would be great if it was tested thoroughly. I couldn't easily measure and compared things (=5.0.4 vs 5.1-dev) - and I feel like I'm spending too much time with watching out for regressions, need to focus on other stuff too.

#12 - 02/25/2015 07:14 PM - Mark Abraham

Szilárd Páll wrote:

Mark Abraham wrote:

Szilárd Páll wrote:

I still see at least 10% performance difference between the bonded in 5.0 and current master. Not even sure if the former bonded f now called listed f cycle-count contains the same thing so backward comparability using the subcounters may not even be impossible.

See commit message for 19fb85c0001928e6de214218bf3047445f056bb7. Position restraints were added, because they should have been there all along.

I assume by "there" means in the counter renamed from "Bonded" to "Listed". Frankly the "should have been there" statement is IMHO questionable and it is something that some of us disagree with it (see comments on change #4461).

Are you seriously suggesting that I should have added a separate subcounter for position restraints to remedy an historical error that time for position restraints was accrued to "Rest"? Nobody does benchmarks on position-restrained runs, so that aspect is immaterial. The only relevant thing discussed at 4461 is that people are scared of lots of (sub)timers for little-used or very fast code.

I was against such that change from the beginning and brought up the very reasons that materialized now: it made comparison unnecessarily hard for the sake of matching the naming of timed entities with aspects of the (new) implementation.

I already said I was happy for the output string to change back. But the commit that does it is bogged in discussion of other issues.

I've asked several times (last #4461) that bonded and other listed interaction counters are kept separate both because (to us) it seems to make more sense and because backward comparability is useful and important. IMO it's not the shared algorithm or module implementation what matters when picking the scope of subcounters and changing the content of counters for no other good reason than to reflect implementation details has obviously created a barrier for cross checking performance against 5.0.

I don't understand what timings you want separate. There's the actual interactions, the reduction of the forces, and nothing else worth timing.

The barrier to checking performance over time is that people want to use diff and simple string matching, not the reorganization of the code. Please separate the two issues. The sub-counter(s) should closely reflect the code organization.

Nothing else should have changed, now that we have restored all of the SIMD support.

The code got reorganized too, did it not? So it would be great if it was tested thoroughly. I couldn't easily measure and compared things (=5.0.4 vs 5.1-dev) - and I feel like I'm spending too much time with watching out for regressions, need to focus on other stuff too.

One of the issues here is that we had several refactoring patches that sat around for months with little or new review (or timely follow-up when the comments were addressed), and meanwhile we did related performance enhancements on a release branch. This combination must stop. Since we must have clean-up, and I can't make people review code, I think I will stop reviewing minor performance enhancements on release branches :)

In the absence of time (or the dedicated machines that I asked for) to build machinery for reliable per-commit timing testing, the occasion for doing the bulk of performance testing is during the pre-release phase that we have not yet begin. I did do some manual performance checking during the listed-forces reorganization (some noted on gerrit), but the moving targets made such work not very useful.

- Related to Bug #1686: bonded force regression added

Teemu Murtola wrote:

gcc has -Wundef to warn about using an undefined preprocessor symbol in an #if. With suitably structured #defines and #if instead of #ifdef, this should also be able to catch these issues. Not sure how many bogus warnings it would currently generate, though.

Could work, perhaps with some white- & black-listing of names. Perhaps best left until after we have new kernel generation scheme, and/or eliminating any other such use of the preprocessor.

I have also seen a French(?) project that aimed at some kind of analysis of preprocessor symbol usage with the objective of eliminating security flaws from the various kinds of associated fails, but can't remember the name.

What would you black/whitelist? (that can be tricky to do, unless we create a separate script that does this, instead of doing this as a byproduct of the compilation). I quickly tested compiling with -Wundef, and after fixing two issues in commonly used headers, non-test code compiles quite nicely (and finds at least one clear issue), so it shouldn't be impossible to make this approach work.

PS. The one clear issue is that simd_math.h uses GMX_SIMD_MATH_TARGET_SINGLE_BITS, which is never defined, so the extra two iterations in gmx_simd_invsqrt_pair_d are always done, irrespective of any target bit counts.
Teemu Murtola wrote:

Mark Abraham wrote:

Teemu Murtola wrote:

gcc has -Wundef to warn about using an undefined preprocessor symbol in an #if. With suitably structured #defines and #if instead of #ifdef, this should also be able to catch these issues. Not sure how many bogus warnings it would currently generate, though.

Could work, perhaps with some white- & black-listing of names. Perhaps best left until after we have new kernel generation scheme, and/or eliminating any other such use of the preprocessor.

What would you black/whitelist? (that can be tricky to do, unless we create a separate script that does this, instead of doing this as a byproduct of the compilation). I quickly tested compiling with -Wundef, and after fixing two issues in commonly used headers, non-test code compiles quite nicely (and finds at least one clear issue), so it shouldn't be impossible to make this approach work.

Sounds good. I was afraid of lots of nearly-false-positive issues, when I suggested lists.

PS. The one clear issue is that simd_math.h uses GMX_SIMD_MATH_TARGET_SINGLE_BITS, which is never defined, so the extra two iterations in gmx_simd_invsqrt_pair_d are always done, irrespective of any target bit counts.

Lovely!

---

#18 - 06/05/2015 08:13 AM - Gerrit Code Review Bot

Gerrit received a related patchset '7' for Issue #1673.
Uploader: Roland Schulz (roland@rschulz.eu)
Change-Id: ldcdf3872b5a169e8690721bbe83922a4ab280da8
The fixes we wanted for 5.1 are in, but I'll leave this open since we should move all our cmakedefines to 0/1 and swap #ifdef for #if when checking configurations.

Applied in changeset 3df2b06e2896161e5f501950b6a065e2375559e2.

issue was automatically marked as resolved but re-opening as per Erik's request

Made a Redmine task for the parent issue, so this can close

Gerrit received a related patchset '1' for Issue #1673.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-id: i59b68747399d528603a86e06ab9bcadd67fd3fa
Gerrit URL: https://gerrit.gromacs.org/5343