### Description
The SIMD reduce() function in double precision returns a float instead of a double with AVX_256 and AVX2_256 several SIMD implementations. This is only used in the PME force gather and thus lead to the PME grid forces only having single precision accuracy.

Also dotProduct is affected by the same cut-and-paste bug, but that is never used in double precision.

Another serious issue is that many SIMD tests use input that only have a few significant bits and therefore returning a single precision result is not caught.

### Related issues:
- Related to GROMACS - Bug #2156: EwaldUnitTests fail on current master using `-DGMX_DOUBLE=ON` added
- Related to GROMACS - Feature #2163: Use better input/output value choices for SIMD tests added

### Associated revisions
**Revision 8b1ee283 - 04/21/2017 06:37 PM - Berk Hess**
Fix bugs in double AVX_256 Simd4

The double precision version of reduce() and dotProduct() returned a float with AVX_256 and AVX2_256.

Only reduce() is used in double, in the PME force gather.

Fixes #2162.

Change-Id: laa44921215e726726b2da223f3677c8637bc65ae

**Revision 30c6a0a0a - 05/14/2017 04:11 PM - Berk Hess**
Fix bugs in double Simd4 implementations

The double precision version of reduce() and dotProduct() returned a float with AVX_128_FMA, AVX_512, MIC and IBM_QPX.

Only reduce() is used in double, in the PME force gather.

The same bugs were fixed before for AVX_256.

Refs #2162.

Change-Id: l1a5f044fc70ba1e8d7928a61e4e2c3d170a3922

### History
#### #1 - 04/21/2017 04:04 PM - Gerrit Code Review Bot
Gerrit received a related patchset ‘2’ for Issue #2162.
Uploader: Berk Hess (hess@kth.se)
Change-Id: gromacs~release-2016~Iaa44921215e726726b2da223f3677c8637bc65ae
Gerrit URL: [https://gerrit.gromacs.org/6598](https://gerrit.gromacs.org/6598)

#### #2 - 04/21/2017 04:05 PM - Aleksei lupinov
- Related to Bug #2156: EwaldUnitTests fail on current master using `-DGMX_DOUBLE=ON` added

#### #3 - 04/21/2017 04:06 PM - Berk Hess
- Related to Feature #2163: Use better input/output value choices for SIMD tests added
As discussed offline while looking for the bug, this seems like a case that should be caught by a static analyzer, but the current setup clang analyzer is only done with: -DGMX_GPU=OFF -DGMX_OPENMP=OFF -DGMX_SIMD=None -DGMX_DOUBLE=OFF. Are there any technical limitation that prevent testing more of the relevant code-paths (especially OpenMP, SIMD (=Reference), and Double)?

As discussed offline while looking for the bug, this seems like a case that should be caught by a static analyzer, but the current setup clang analyzer is only done with: -DGMX_GPU=OFF -DGMX_OPENMP=OFF -DGMX_SIMD=None -DGMX_DOUBLE=OFF. Are there any technical limitation that prevent testing more of the relevant code-paths (especially OpenMP, SIMD (=Reference), and Double)?

Not that I know of (and if there are, they may no longer apply). Trying alternatives in admin/builds/clang-analyzer.py should be straightforward.

Slightly OT: one quite significant limitation for my use-case is that I seem to be unable to even get an analyzer build to run on my desktop. For some reason c++-analyzer it keeps trying to use g++ regardless of whether I pass c++-analyzer as CXX compiler or prefix the cmake invocation with scan-build. What am I missing?

Slightly OT: one quite significant limitation for my use-case is that I seem to be unable to even get an analyzer build to run on my desktop. For some reason c++-analyzer it keeps trying to use g++ regardless of whether I pass c++-analyzer as CXX compiler or prefix the cmake invocation with scan-build. What am I missing?

Don't know offhand, but the docs are at
Not that I know of (and if there are, they may no longer apply). Trying alternatives in admin/builds/clang-analyzer.py should be straightforward.

Slightly OT: one quite significant limitation for my use-case is that I seem to be unable to even get an analyzer build to run on my desktop. For some reason c++-analyzer it keeps trying to use g++ regardless of whether I pass c++-analyzer as CXX compiler or prefix the cmake invocation with scan-build. What am I missing?

Don’t know offhand, but the docs are at

releng/environment.py has other relevant things, so perhaps the update Vedran made doesn’t go far enough yet

#9 - 04/21/2017 11:39 PM - Szilárd Páll

Mark Abraham wrote:

Don’t know offhand, but the docs are at

Looked at that as well as the exact cmake invocation in the jenkins output, but I’m either missing something or there is something custom hard-coded on the slave as it seems to not work with either repo-based clang install nor with custom builds.

#10 - 04/21/2017 11:43 PM - Szilárd Páll

Mark Abraham wrote:

releng/environment.py has other relevant things, so perhaps the update Vedran made doesn’t go far enough yet

Indeed, just noticed that it also sets CCC_CC and CCC_CXX env vars which is not something even on the scan-build docs page was not present (and --use-cc didn’t really work).

#11 - 04/22/2017 12:17 AM - Mark Abraham

Szilárd Páll wrote:

releng/environment.py has other relevant things, so perhaps the update Vedran made doesn’t go far enough yet

Indeed, just noticed that it also sets CCC_CC and CCC_CXX env vars which is not something even on the scan-build docs page was not present (and --use-cc didn’t really work).

Just in case it’s useful in the short term, I got an analysis that clearly succeeded in finding an issue with approximately

    export CCC_CXX=/usr/bin/clang-3.9
    cmake -G Ninja -DCMAKE_INSTALL_PREFIX=install -DGMX_MPI=off -DGMX_THREAD_MPI=on -DGMX_OPENMP=off -DGMX_DOUBLE=on 
    -DREGRESSIONTEST_PATH=/home/marklocal/git/regressiontests -DCMAKE_BUILDTYPE=Debug -DCMAKE_C_COMPILER=clang 
    -DCMAKE_BUILD_TYPE=Debug -DCMAKE_CXX_COMPILER=/usr 
    /lib/clang/c++-analyzer -DGMX_STDLIB_CXX_FLAGS=-stdlib=libc++ -DGMX_STDLIB_LIBRARIES="*-lc++abi;*-lc++" 
    scan-build ninja

#12 - 04/22/2017 09:20 AM - Berk Hess

- Status changed from In Progress to Resolved

Applied in changeset 8b1ee2835b80e6c6f9ee64f8d0b0170ea46b83c.

#13 - 04/27/2017 07:28 PM - Szilárd Páll

Mark Abraham wrote:

Just in case it’s useful in the short term, I got an analysis that clearly succeeded in finding an issue with approximately

For the record: I did indeed succeed by using the aforementioned CCC_CC/CCC_CXX variables. This is the magic that makes it work but the use of env vars is unfortunately not logged by the releng scripts so such things are hard to repro outside of jenkins.
Gerrit received a related patchset ‘1’ for Issue #2162.
Uploader: Berk Hess (hess@kth.se)
Change-Id: gromacs~release-2016~If1a5f044fc70ba1e8d7928a61e4e2c3d170a3922
Gerrit URL: https://gerrit.gromacs.org/6618

Gerrit received a related patchset ‘1’ for Issue #2162.
Uploader: Mark Abraham (mark.j.abraham@gmail.com)
Change-Id: gromacs~release-2016~Ib5a749b4dee205679d4b2fc6a830e0d6e20f0007
Gerrit URL: https://gerrit.gromacs.org/6619

- Subject changed from AVX(2) double precision reduce is actual single precision to Several SIMD4 double precision reduce are actual single precision
- Description updated

- Status changed from Resolved to Closed