Erik and I have some plans to test the BlueGene/Q processor (since we suspect the guarantee of 5 bits of precision is too conservative) and (at least) use that for the kernel use of rsqrt functionality. More general use within the GROMACS codebase and to general PowerPC contexts is awkward, because we can't test all PowerPC contexts.

Note that gcc on PowerPC with relaxed maths settings will compile 1/sqrt(x) to the frsqrte + Newton-Raphson solution and (IIRC) implements rsqrt() the same way. Presumably xlc is similar. We don't want relaxed maths generally, though, we just know some places where the accuracy of gmx_software_invsqrt is OK.

On PowerPC, maybe setting GMX_SOFTWARE_INVSQRT=FALSE, so gmx_invsqrt gets #defined to rsqrt() will work - we introduce code pragmas where we know relaxed accuracy is acceptable, and let the compiler take care of the details.

Associated revisions
Revision 690bfd41 - 12/15/2015 12:09 PM - Erik Lindahl
C++ math function cleanup

math/functions.h now implements a number of old and new math functions with either float, double, or integer arguments. Manual SIMD versions of 1/sqrt have been tested with gcc and icc on x86, Power8, Arm32 and Arm64, but with correct 'f' suffixes on constants there is only 10-15% performance difference, so for now we always use the system versions to avoid having this file depend on config.h. Functions for third and sixth roots have been introduced to replace many of our pow() calls, and the code has been cleaned up to use the new functions.

Refs #1111.

Change-Id: I74340987fff68bc70d268f07dbddf63eb706db32

History
#1 - 01/09/2013 03:46 PM - Mark Abraham
- Category set to mdrun

Follows on from #947.

#2 - 01/10/2014 08:00 PM - Szilárd Páll
Bump. Has this been sorted out?

#3 - 01/10/2014 08:21 PM - Erik Lindahl
Almost - my new SIMD code uses explicit defines for the number of bits of accuracy provided by the instruction set, and then adjusts the number of iterations.

In theory we could add a test for the accuracy during cmake configuration, but the problem is that the only architecture where this could be a problem right now enforces cross-compiles. A better way might be to introduce a sanity check at the start of execution that tests the rsqrt() accuracy for 1-2 sample random values (and likely some other config stuff too) to make sure they are in line with the things specified during configuration.

#4 - 01/10/2014 08:23 PM - Erik Lindahl
PS: As far as I know, table lookup + N-R iterations is the only efficient numerical algorithm to compute either sqrt or 1/x. The only difference is the amount of hardware (or microcode) support, and how much the implementation checks for zeros, denormal values, +infinity, and negative numbers.
Mark Abraham wrote:

We don't want relaxed maths generally, though, we just know some places where the accuracy of gmx_software_invsqrt is OK.

What do you mean by relaxed math? Accepting more than 1 ulps error or non IEEE compliant math (something like the gcc options fassociative-math/funsafe-math-optimizations/ffast-math)?

For 1/sqrtf(x)

- ICC15 always produces vrsqrtss
- GCC49 produces vrsqrtss with "-ffast-math -O3" or "-O3 -unsafe-math-optimizations -finite-math-only" (oddly specifying the options corresponding to funsafe-math-optimizations doesn't give the same result) and sqrtss with "-O3"
- clang35 produces sqrtss with "-ffast-math -O3" and "callq sqrtf" with "-O3" Is "-ffast-math" (or some subset) safe to use? If it is safe we should enable it by default for GCC and clang. If not we need to use something like fp-model precise in ICC. Also notice that ICC uses 4upls by default. Is that OK for Gromacs?

After we decided what level of relaxed/unsafe-math is sufficient, we should disable GMX_SOFTWARE_INVSQRT for any compiler which automatically produces efficient rsqrt for that setting.

Roland Schulz wrote:

Just wondering: why not use the SIMD intrinsics directly if these are really faster than GMX_SOFTWARE_INVSQRT?

gmx_invsqrt is only in code which hasn't been vectorized yet. We would have to map it to the scalar version (v)rsqrtss (+iteration). But the compiler might be able to auto-vectorize and map it to packed one (v)rsqrtps. I'm not sure whether any of the auto-vectorizer can convert a scalar intrinsic to a packed one but that seems like making it unnecessary complicated to the compiler, because they obviously know how to map 1/sqrt in an auto-vectorized loop to the packed one.

Gerrit received a related patchset '4' for Issue #1111.
Uploader: Erik Lindahl (erik.lindahl@gmail.com)
Change-Id: I74340987fff68bc70d268fc691f64988e7a3eb706d0b32
Gerrit URL: https://gerrit.gromacs.org/5259

Gerrit received a related patchset '6' for Issue #1111.
Uploader: Erik Lindahl (erik.lindahl@gmail.com)
Change-Id: I7631ef5151b306a4de1d0649ae45e464b9d8a436
Gerrit URL: https://gerrit.gromacs.org/5276

Gerrit received a related patchset '1' for Issue #1111.
Uploader: Mark Abraham (mark@abraham.com)
Change-Id: I5b7f85a1b53d7d986ce16dc35fa90e84d0f50
Gerrit URL: https://gerrit.gromacs.org/5337

- Status changed from New to Closed
- Target version changed from future to 5.1
I'll treat this as solved by the new SIMD layer in 5.1