The task is about renaming of MPI-related conditional-compilation machinery. Initially, Teemu noted that the commit message for a related change should be improved. He suggested considering renaming the GMX_MPI source level defines and changing the way the user provides the GMX_MPI/GMX_THREAD_MPI value. Instead of two separate, interdependent values, something like GMX_DOMAIN_PARALLELIZATION=off/thread-MPI/MPI could be set, and source-level defines based on this. Additionally, renaming the GMX_MPI source level defines was suggested, as they were poorly named and led to proliferation of conditions like 

```
#if defined GMX_MPI && !defined GMX_THREAD_MPI
```

He also mentioned that unrelated to this change, a commit was made to remove #ifdef GMX_THREAD_MPI for basic thread safety. This simplifies the code and reduces the number of #ifdefs. Moreover, the GMX_MPI management in CMake code was changed to clarify the semantics. Instead of having logic in config.h, it was moved to config.c. A related issue is #948.
The semantics of GMX_MPI are still different in the source code compared to CMake, but that cannot be solved before some conclusion is reached in #1320.

There have been several bugs (at least most caught during code review) from assuming that GMX_MPI could be used to access the user input value everywhere in CMake, and it is very confusing to follow the CMake code because of this issue (the same code can have different behavior, depending on the point where it gets executed).

CMake code still mixes GMX_LIB_MPI and GMX_MPI to check for the same thing, but at least that no longer causes bugs. Removed an obsolete if (GMX_MPI AND NOT GMX_THREAD_MPI) construct.

Fix a regex in gmxtree.py to properly parse '#if defined ...' in config.h.

Related to #1320.

Change-Id: Icbdedd30eaa39978b6a302660207376fce1bba4e

---

**History**

**#1 - 08/12/2013 02:03 PM - Mark Abraham**

I generally agree - thread MPI and real MPI are designed to be transparently equivalent or mutually exclusive. The current CMake interface seems to be an artefact of the convenience of option() for a toggle and the lack of CMake support for a proper enumeration.

I don't like the length of GMX_DOMAIN_PARALLELIZATION, despite being a reasonable description. There's some value in maintaining consistency with the current interface.

I suggest:

- CMake variable GMX_MPI that defaults to "thread-MPI," with logic to interpret a user-supplied value of "on" as "external-MPI," and either/both of "off" and "serial" to mean no kind of MPI at all. Teemu once showed me some draft CMake-ry that might help support a proper enumeration here?
- config.h sets GMX_SERIAL, GMX_THREAD_MPI, GMX_MPI accordingly
- config.h maybe sets GMX_ANY_MPI, if we'd prefer that to #ifndef GMX_SERIAL (I don't mind which way we go)
- GMX_LIB_MPI is retired
- perverse #if predicates are fixed accordingly

This seems to me likely to maintain most people's build commands working the same way they do now.
I agree that GMX_DOMAIN_PARALLELIZATION is probably too long; I chose that in my original post to make it easier to understand without a long explanation. I don't have any better proposals, though, so reusing the existing GMX_MPI is a reasonable alternative (as long as we don't add any non-MPI communication alternatives...). It would be simpler to not have multiple accepted values for it, though; if we choose to accept "on", what other options should we accept? CMake does minimal validation on the input, and currently anything that isn't explicitly "off" or "false" or some of the other predetermined alternatives is considered "on". My draft would require some extra complexity to support multiple alternatives that map to the same thing, but I can do that if we want to go that way.

For the code defines, I think if we keep GMX_MPI in the code, then it should keep its current meaning to minimize confusion (and also complexity in CMake, to avoid trying to map GMX_MPI=thread-MPI to undefined GMX_MPI in the code). Other than that, I don't have immediate suggestions for the names.

Teemu Murtola wrote:

I agree that GMX_DOMAIN_PARALLELIZATION is probably too long; I chose that in my original post to make it easier to understand without a long explanation. I don't have any better proposals, though, so reusing the existing GMX_MPI is a reasonable alternative (as long as we don't add any non-MPI communication alternatives...).

No promises there :-)

It would be simpler to not have multiple accepted values for it, though; if we choose to accept "on", what other options should we accept? CMake does minimal validation on the input, and currently anything that isn't explicitly "off" or "false" or some of the other predetermined alternatives is considered "on". My draft would require some extra complexity to support multiple alternatives that map to the same thing, but I can do that if we want to go that way.

The CMake interface expects "on" to have a meaning and the former meaning of GMX_MPI was "on" => use external MPI. I'd like to keep that. We could choose to have a no-nonsense three-value design (e.g. "off", "threads" (the default), "on")

For the code defines, I think if we keep GMX_MPI in the code, then it should keep its current meaning to minimize confusion (and also complexity in CMake, to avoid trying to map GMX_MPI=thread-MPI to undefined GMX_MPI in the code). Other than that, I don't have immediate suggestions for the names.

There's a conflict of principles between keeping the meaning of the code #define GMX_MPI, and losing the confusion about GMX_LIB_MPI. I'm happy either way, so long as we get rid of silly complex predicates.
#4 - 08/12/2013 06:49 PM - Roland Schulz
GMX_DD would be an option for a shorter name than GMX_DOMAIN_PARALLELIZATION. And DD is already used as an abbreviation in other places (e.g. mdrun -dd) so it might be relative easy to understand.

#5 - 08/12/2013 07:57 PM - Teemu Murtola
Mark Abraham wrote:

The CMake interface expects "on" to have a meaning and the former meaning of GMX_MPI was "on" => use external MPI. I'd like to keep that. We could choose to have a no-nonsense three-value design (e.g. "off", "threads" (the default), "on")

The only place in CMake that places any special meaning to "on" is in the GUI/in ccmake. Our installation instructions may also do that, but the user could just as well give -DGX_MPI=1, -DGX_MPI=TRUE, or even -DGX_MPI=foobar on the command-line, and all would lead to MPI getting enabled. So my point was that if we make GMX_MPI a multichoice option, we need to explicitly enumerate which of these (unlimited) alternatives we now accept and which we reject (since we definitely don't want a typo like GMX_MPI=thread-MPI to silently enable MPI...). Another alternative would be to put in a bit of backward compatibility glue: keep GMX_MPI as a (deprecated) option, and if it is set, set the default value for the new option (whether it be GMX_DOMAIN_PARALLELIZATION or something else) accordingly. This would probably also be easier to implement.

For the code defines, I think if we keep GMX_MPI in the code, then it should keep its current meaning to minimize confusion (and also complexity in CMake, to avoid trying to map GMX_MPI=thread-MPI to undefined GMX_MPI in the code). Other than that, I don't have immediate suggestions for the names.

There's a conflict of principles between keeping the meaning of the code #define GMX_MPI, and losing the confusion about GMX_LIB_MPI. I'm happy either way, so long as we get rid of silly complex predicates.

I don't see what the conflict is. If we change the meaning of GMX_MPI in the code, there is a great risk that a merge (whether from release-4-6, or from a feature branch developed before the switch) forgets to change some defines, and causes hard-to-find issues. If we instead remove GMX_MPI completely from the code, such mistakes could be easily found by just grepping for GMX_MPI. Leaving GMX_MPI as it is now is also an option, but that is somewhat confusing if there is also a GMX_MPI CMake option that does something else. So I would perhaps lean towards renaming both GMX_MPI and GMX_LIB_MPI to something else.
Teemu Murtola wrote:

Mark Abraham wrote:

The CMake interface expects "on" to have a meaning and the former meaning of GMX_MPI was "on" => use external MPI. I'd like to keep that. We could choose to have a no-nonsense three-value design (e.g. "off", "threads" (the default), "on")

The only place in CMake that places any special meaning to "on" is in the GUI/in ccmake. Our installation instructions may also do that, but the user could just as well give -DGMX_MPI=1, -DGMX_MPI=TRUE, or even -DGMX_MPI=foobar on the command-line, and all would lead to MPI getting enabled. So my point was that if we make GMX_MPI a multichoice option, we need to explicitly enumerate which of these (unlimited) alternatives we now accept and which we reject (since we definitely don't want a typo like GMX_MPI=tread-MPI to silently enable MPI...).

Yep.

Another alternative would be to put in a bit of backward compatibility glue: keep GMX_MPI as a (deprecated) option, and if it is set, set the default value for the new option (whether it be GMX_DOMAIN_PARALLELIZATION or something else) accordingly. This would probably also be easier to implement.

OK, GMX_DD is nice and quick to type. GMX_PARALLEL is perhaps a bit more accessible to people who don't know what DD is (sysadmins). Technically OpenMP, CUDA and SIMD are all forms of parallelism too, but there is more of a historical expectation that parallel means MPI.

For the code defines, I think if we keep GMX_MPI in the code, then it should keep its current meaning to minimize confusion (and also complexity in CMake, to avoid trying to map GMX_MPI=thread-MPI to undefined GMX_MPI in the code). Other than that, I don't have immediate suggestions for the names.

There's a conflict of principles between keeping the meaning of the code #define GMX_MPI, and losing the confusion about GMX_LIB_MPI. I'm happy either way, so long as we get rid of silly complex predicates.

I don't see what the conflict is. If we change the meaning of GMX_MPI in the code, there is a great risk that a merge (whether from release-4-6, or from a feature branch developed before the switch) forgets to change some defines, and causes hard-to-find issues. If we instead remove GMX_MPI completely from the code, such mistakes could be easily found by just grepping for GMX_MPI. Leaving GMX_MPI as it is now is also an option, but that is somewhat confusing if there is also a GMX_MPI CMake option that does something else. So I would perhaps lean towards renaming both GMX_MPI and GMX_LIB_MPI to something else.

Good point with the merging. How about GMX_ANY_MPI and GMX_EXTERNAL_MPI for the code #defines?
Mark Abraham wrote:

Good point with the merging. How about GMX_ANY_MPI and GMX_EXTERNAL_MPI for the code #defines?

That's one option. Another would be to consider what are really the differences between thread MPI and real MPI, and name the defines accordingly (e.g., GMX_USE_MPI_COMMUNICATION for GMX_MPI and something else for initialization). After https://gerrit.gromacs.org/#/c/2600/, there are now ~40 #ifdef GMX_THREAD_MPI and ~15 #ifdef GMX_LIB_MPI blocks in the code, so I'm not sure whether it is easy to find a common denominator for all of those, though. Mostly these seem to be related to different initialization; some may be unnecessary. Some GMX_THREAD_MPI occurrences are completely unrelated to MPI itself, so those can probably remain if necessary.

#8 - 09/23/2013 09:13 PM - Szilárd Páll

I have a few comments to add:

- Calling mdrun "serial" is not appropriate because this is only true if OpenMP, thread-MPI, MPI (and SIMD/GPU kernels) are all disabled which should be extremely rare. The age of single-core architecture is over and there are very few platforms where some sort of multi-threading or MPI won't work. Hence, oversimplification by GMX_SERIAL is not a good idea, IMHO. Similarly, GMX_PARALLEL can also lead to confusion.

- Domain-decomposition and the actual parallelization scheme that implements it are closely related, but I'm not sure that it's a good idea to require the user to know about such details. Additionally, most people know that MPI is the way to run "parallel" on a cluster and by now most should have gotten used to the fact that "-nt" works by default on non-MPI builds. Adding a GMX_DD with enumerated values would potentially result in confusion as users won't know anymore how to enable MPI. On the other hand, keeping GMX_MPI (and GMX_THREAD_MPI) for user convenience next to GMX_DD={thread-MPI, MPI, off} would mean extra CMake code - although not very complicated. Hence, while I find GMX_DD OK for advanced users, in many cases will hurt especially if GMX_MPI gets removed.

- There is a strong need for other inter-node communication mechanisms, in particular lower-level communication libraries for one-sided and RMDA communication. There is ongoing work on this, but it is unclear when will it make it into a release. However, if the naming scheme is modified, it would be good to leave room for such new features to be easily added to the build system.

- thread-MPI was designed to be a drop-in replacement for the MPI functionality in GROMACS, so AFAIK there are very few cases (if any) except (un)initialization where a distinction between MPI and thread-MPI needs to be made. On the other hand, there will always be a need to distinguish between the DD/no DD case as well as MPI vs thread-MPI, so using GMX_DD together with GMX_EXTERNAL_MPI and GMX_THREAD_MPI sound to me like a decent solution.

#9 - 09/24/2013 09:39 AM - Mark Abraham

Szilárd Páll wrote:
I have a few comments to add:

- Calling mdrun "serial" is not appropriate because this is only true if OpenMP, thread-MPI, MPI (and SIMD/GPU kernels) are all disabled which should be extremely rare. The age of single-core architecture is over and there are very few platforms where some sort of multi-threading or MPI won't work. Hence, oversimplification by GMX_SERIAL is not a good idea, IMHO. Similarly, GMX_PARALLEL can also lead to confusion.

Sure, but we will continue to maintain a "no MPI" and a "no serial" build if only to support the Reference build type / for testing. I agree that calling the "no MPI" define GMX_NO_MPI would be preferable.

- Domain-decomposition and the actual parallelization scheme that implements it are closely related, but I'm not sure that it's a good idea to require the user to know about such details. Additionally, most people know that MPI is the way to run "parallel" on a cluster and by now most should have gotten used to the fact that "-nt" works by default on non-MPI builds. Adding a GMX_DD with enumerated values would potentially result in confusion as users won't know anymore how to enable MPI. On the other hand, keeping GMX_MPI (and GMX_THREAD_MPI) for user convenience next to GMX_DD=\{thread-MPI, MPI, off\} would mean extra CMake code - although not very complicated. Hence, while I find GMX_DD OK for advanced users, in many cases will hurt especially if GMX_MPI gets removed.

- There is a strong need for other inter-node communication mechanisms, in particular lower-level communication libraries for one-sided and RMDA communication. There is ongoing work on this, but it is unclear when will it make it into a release. However, if the naming scheme is modified, it would be good to leave room for such new features to be easily added to the build system.

Do we actually want a forest of booleans after all? That has the virtue of being easily extensible (e.g. GMX_USE_IVERBS, GMX_USE_RDMA, GMX_USE_SHMEM). I think the current logic supporting the mutual exclusivity of booleans GMX_USE_EXTERNAL_MPI and GMX_USE_THREAD_MPI is acceptable - because we do not cache either, if either is set then the other cannot be set. Note that the names in the CMake interface do not have to correlate strongly with the #defines it will trigger.

I do feel strongly that new CMake boolean options should be verb-like, and preferably use a positive sense. So, GMX_ENABLE_XXX or GMX_USE_XXX rather than GMX_DISABLE_XXX.

- thread-MPI was designed to be a drop-in replacement for the MPI functionality in GROMACS, so AFAIK there are very few cases (if any) except (un)initialization where a distinction between MPI and thread-MPI needs to be made.

Even after Teemu's [https://gerrit.gromacs.org/#/c/2600/](https://gerrit.gromacs.org/#/c/2600/) there's still ~50 preprocessor MPI predicates, mostly in low-level or setup code. By the time we hit the main MD loop, however, there should be few/no differences in the code path.

On the other hand, there will always be a need to distinguish between the DD/no DD case as well as MPI vs thread-MPI, so using GMX_DD together with GMX_EXTERNAL_MPI and GMX_THREAD_MPI sound to me like a decent solution.

Hmm. "no DD" means "use no MPI even if it exists," so turning the MPI options off implies "no DD." I don't yet see the use case for a separate GMX_USE_DD, e.g. so that it can be off while some kind of MPI (for analysis tools?) is on. I think the proposal for something like GMX_DD was to be a three-way replacement for the CMake GMX*MPI booleans, not to augment them.
#10 - 06/20/2014 10:42 AM - Erik Lindahl
- Target version changed from 5.0 to 5.x

#11 - 01/04/2015 03:59 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1320.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: Icbdedd30eaa39978b6a302660207376f6ce1bba4e
Gerrit URL: https://gerrit.gromacs.org/4346

#12 - 01/04/2015 03:59 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1320.
Uploader: Teemu Murtola (teemu.murtola@gmail.com)
Change-Id: Icbdedd30eaa39978b6a302660207376f6ce1bba4e
Gerrit URL: https://gerrit.gromacs.org/4346