Gromacs - Bug #1543
complex/nbnxn_vsite regression tests fails with GPUs

06/30/2014 11:03 PM - Szilárd Páll

Status: Closed
Priority: Normal
Assignee: 
Category: testing
Target version: 5.0.1
Affected version - extra info:

Affected version: 5.0

Description
The complex/nbnxn_vsite test fails when mdrun attempts to use GPUs because apparently gmxtest.pl now explicitly starts N ranks (N=6?).

This is probably caused by the parallelism limitation feature in the regressiontests script introduced in e5827da.

Related issues:
Related to Gromacs - Bug #1550: GMX_MAX_MPI_THREADS has no implementation Closed 07/02/2014
Related to Gromacs - Bug #1624: complex/nbnxn_vsite regression test fails when... Closed 10/10/2014

Associated revisions
Revision 35e2d2c8 - 07/03/2014 06:05 PM - Mark Abraham
Fix and improve behaviour for retrying mdrun

Implemented use_separate_pme_ranks in terms of the number of MPI ranks. Fixed a bug where it would leave undef in the number of PP ranks in use, which stopped -gpu_id being propagated.

Added support to the mdrun-retry mechanism for adapting the number of ranks to equal the number of GPUs when the first attempt failed.

Added output to inform the user explicitly that a retry is being attempted.

Added new fatal error if the user specifies both -nt and -np, which supports the existing assumptions that only one kind of MPI-level parallelism is being attempted.

When reporting the style of mdrun calls that might be made, some of the output now acknowledges that the harness might be smart and make re-attempts with different settings than the user/default specified.

Improved documentation here also.

Fixes #1543

Change-Id: i.e36fc26b72fac3a90b6b6991533fb0913c6afcb
The regression tests script worked so far automatically with GPUs and thread-MPI because mdrun automatically starts the right number of ranks required by the number of GPUs available. However, now, the rank limit gets always set unless the user does not set the number of ranks. To keep the script running with GPUs and thread-MPI we should use the GMX_MAX_MPI_THREADS environment variable instead of setting -ntmpi.

My perl knowledge is limited, but I'll try to fix it tomorrow.

Right. Sorry about that - I was fixing the other "nice" automatic feature of spawning a thread-MPI rank per core. Apparently Jenkins specifies enough to keep things happy.

To keep the script running with GPUs and thread-MPI we should use the GMX_MAX_MPI_THREADS environment variable instead of setting -ntmpi.

Sounds like a nice solution, but after searching the git history, GMX_MAX_MPI_THREADS has never had an implementation in public branches of GROMACS. Rather than setting -ntmpi, setting -nt probably copes better with the tMPI+GPU case, while doing no damage to any other case. How does that sound instead?

My perl knowledge is limited, but I'll try to fix it tomorrow.

I've promised some more cleanup of this machinery (long overdue), and I can do this then also.
The regression tests script worked so far automatically with GPUs and thread-MPI because mdrun automatically starts the right number of ranks required by the number of GPUs available. However, now, the rank limit gets always set unless the user does not set the number of ranks.

Right. Sorry about that - I was fixing the other "nice" automatic feature of spawning a thread-MPI rank per core. Apparently Jenkins specifies enough to keep things happy.

Jenkins must always specify -ntmpi, otherwise this would have failed on the GPU boxes.

To keep the script running with GPUs and thread-MPI we should use the GMX_MAX_MPI_THREADS environment variable instead of setting -ntmpi.

Sounds like a nice solution, but after searching the git history, GMX_MAX_MPI_THREADS has never had an implementation in public branches of GROMACS.

Weird. I am pretty sure GMX_MAX_MPI_THREADS did exist at some point. The idea was to allow users to set this variable e.g. in their .bashrc in order to limit the number of tMPI threads spawned automatically without having to always type in the command line option. This should be useful for people who use their laptop/workstation e.g. for minimization or equilibration, but want to leave 1-2 cores available for other tasks.

I thought Berk used this on his desktop machine, but apparently there is only some leftover code in gmx_detect_hardware.c indicating that at some point it must have been used.

Rather than setting -ntmpi, setting -nt probably copes better with the tMPI+GPU case, while doing no damage to any other case. How does that sound instead?

-nt does not serve the same purpose as GMX_MAX_MPI_THREADS (would/did), the former sets the total number of threads rather than imposing a maximum. While this should not be a problem with current limit of 6 only used in a single test (as this will oversubscribe hardware only in a few cases), if we want to adopt this mechanism for other tests too, it using -nt is not a good solution. A much larger limit would certainly cause trouble if e.g. we start 64 threads for some test on a 4-core machine.

So, to summarize, as a temporary fix it should be fine to set -nt, but perhaps it's best if we simply re-introduce the GMX_MAX_MPI_THREADS variable.

My perl knowledge is limited, but I'll try to fix it tomorrow.

I've promised some more cleanup of this machinery (long overdue), and I can do this then also.

That would be nice. I gave up yesterday after tinkering with perl for a half our - even before realizing that GMX_MAX_MPI_THREADS doesn't do a thing.
PS: Somewhat unrelated, but running mdrun -nt N leads to an error when N % N_GPU != 0, so a (hypothetical) -nt 5 will rarely work :P. I'm quite sure that this used to work correctly when I implemented it, so it must have broken during the 4.6 "patch" release process - perhaps when the detection code got shuffled around around 4.6.2-3. Will file a separate report.

Wrong, -nt 5 will work only if there is a single or five GPUs in the machine.

-nnt does not serve the same purpose as GMX_MAX_MPI_THREADS (would/did), the former sets the total number of threads rather than imposing a maximum. While this should not be a problem with current limit of 6 only used in a single test (as this will oversubscribe hardware only in a few cases), if we want to adopt this mechanism for other tests too, it using -nt is not a good solution. A much larger limit would certainly cause trouble if e.g. we start 64 threads for some test on a 4-core machine.

The limit's intended to help make sure that by default regressiontests doesn't start with DD that is bound to fail on the test cases that we design to be small. Enforcing the limit with -nt instead of -ntmpi lets the GPU code decide that the result can be however many ranks suit the GPUs, which seems to solve the issue for now. Solving the general case (e.g. max 4 threads, but 3 GPU available) would also require adapting to GPU-related error messages, which I can probably also add.

We seem unlikely to ever have a test case in regressiontests that is large enough to consider running with 64 CPU threads, so I'm not sure what point you're making there. We need a limit so that on a machine with large numbers of cores, the regressiontests can just work; same goes for GPU machines, of course.

So, to summarize, as a temporary fix it should be fine to set -nt, but perhaps it's best if we simply re-introduce the GMX_MAX_MPI_THREADS variable.

My perl knowledge is limited, but I'll try to fix it tomorrow.

I've promised some more cleanup of this machinery (long overdue), and I can do this then also.

That would be nice. I gave up yesterday after tinkering with perl for a half our - even before realizing that GMX_MAX_MPI_THREADS doesn't do a thing.
Mark Abraham wrote:

The limit's intended to help make sure that by default regression tests doesn't start with DD that is bound to fail on the test cases that we design to be small. Enforcing the limit with -nt instead of -ntmpi lets the GPU code decide that the result can be however many ranks suit the GPUs, which seems to solve the issue for now. Solving the general case (e.g. max 4 threads, but 3 GPU available) would also require adapting to GPU-related error messages, which I can probably also add.

-nt N does not impose a limit, it tells mdrun to use exactly N threads. Now, as machines with large number of hardware threads exist, it would be useful to impose limits on more/all tests not only on the current four, isn't it?

However, putting e.g. 32 in the max-mpi-processes file would mean that the respective tests will always run with 32 threads, even if only two are available. Moreover, always running tests with the same number of threads can even hide bugs.

So my point was that if the max-mpi-processes file is not just a temporary hack, we can't simply set -nt, but we need a correct way to impose a maximum.

Additionally, the naming could be improved too. It's not the number of processes that we want to limit, but the number of PP ranks. Besides the ranks/processes nitpick this is relevant because at some point we will hopefully run regression tests with separate PME ranks too.

Mark Abraham wrote:

Actually, I now recall that it is necessary that gmxtest.pl recognize an error message before it may re-run, which I have now implemented for this case.

In any case, I tried using -nt 6 in gmxtest.pl and it did not do the kind of thing we'd like for nbnxn_vsite defaulting to six threads (ie did not split into 2 tMPI ranks if there are 2 GPUs detected). So we can forget about my thought that -nt would permit a hack solution.

-nt N does not impose a limit, it tells mdrun to use exactly N threads. Now, as machines with large number of hardware threads exist, it would be useful to impose limits on more/all tests not only on the current four, isn't it?
Yes. Currently this is done in gmxtest.pl by observing DD fail and re-running with 8 ranks. The present issue was not a DD failure, so there was no rerun. We have lately reverted to re-running only in response to a particular known failure, so we will need to gradually add clauses such as at https://gerrit.gromacs.org/#/c/3747/; old behaviour (rerun, guessing stuff randomly) was poorly stable, understandable, etc.

However, putting e.g. 32 in the max-mpi-processes file would mean that the respective tests will always run with 32 threads, even if only two are available. Moreover, always running tests with the same number of threads can even hide bugs.

All attempts must respect that maximum, and by default it picks that maximum. Then the normal rerun mechanism acts. There's nothing that requires any test to run with a particular number of anything, unless it is intrinsic to the structure of the test (e.g. no support for DD).

So my point was that if the max-mpi-processes file is not just a temporary hack, we can't simply set -nt, but we need a correct way to impose a maximum.

The new mechanism for imposing a maximum on the number of threads works correctly - the issue at hand is that it does not also cooperate with the default of mdrun to issue a fatal error (here, when GPUs are present and the number of ranks is not specified) and the new policy that gmxtest.pl may attempt a rerun only in response to a known error.

Additionally, the naming could be improved too. It's not the number of processes that we want to limit, but the number of PP ranks.

Yes, that was on my list to fix also. Now at https://gerrit.gromacs.org/#/c/3746/1

Besides the ranks/processes nitpick this is relevant because at some point we will hopefully run regression tests with separate PME ranks too.

We've been doing that for a couple of months, see https://gerrit.gromacs.org/#/c/3224/ and Gromacs_Gerrit_5_0 configuration.
Fixed in/before https://gerrit.gromacs.org/#/c/3747/

#11 - 10/10/2014 02:58 AM - Szilárd Páll
- Related to Bug #1624: complex/nbmxn_vsites regressiontest fails when doing CPU-only rerun added