Dear Gromacs developers,

I have been asked to open an issue regarding my observation:
I reproducibly have significantly different results when using Gromacs 5.x with GPUs compared to cpu-only runs. In contrast, I have similar results with GPU and CPU-only runs using Gromacs 4.6.x (that are also similar to the cpu-only results using version 5.x).

For example, with Gromacs 5.x using GPUs, I have lower total energy and higher densities (illustrated by the plots in the pdf files in the tarball).

The tarball attached to this message contains further details in "fileinfo.txt". Included are 4 simulations of a DMPC membrane in NaCl solution using gromacs 4.6.5 and 5.0.1 either with or without two GPUs (GTX 780 Ti) on a single node (and all the logfiles, output files (excluding the trajectory), structure files and topologies in order to reproduce the observation).

The Gromacs compilations (using gcc-4.8.2, fftw-3.3.4, cuda-6.5) always passed the regression test.

Kind regards,
Jan-Philipp Machtens (j.machtens@fz-juelich.de)
Institute of Complex Systems - Cellular Biophysics (ICS-4)
Forschungszentrum Jülich, Germany

Associated revisions
Revision 8e974b0e - 09/30/2014 08:49 PM - Szilárd Páll
Fix incorrect LJ cut-off with GPU + PME tuning

Due to the mismatch of the macro used in generating and implementing the twin cut-off CUDA kernels (used with PP-PME load balancing), the VdW cut-off check was not generated and the (larger) electrostatics cut-off was enforced instead, causing incorrect results with PME tuning.

Fixes #1603
Change-Id: I43ae19968b30843cfe407e927a5cf0bd35c62881

History
#1 - 09/23/2014 06:34 PM - Mark Abraham
Thanks for the comprehensive report - it does seem like the code has changed and not for the better (e.g. step 0 energy with GPU is fine, but after PME tuning the potential energy has increased by 1.5e4 kJ/mol). For now, I would suggest running mdrun -notunepme and see how you go.
Yes, exactly!
The different energies with Gromacs 5.x with GPUs compared to cpu-only runs may be related to the CPU/GPU load balancing.
With "mdrun -notunepme" Gromacs 5.x always (i.e. with and without GPUs) gives gives the "expected" results!
Best, Jan-Philipp

I have also observed that the issue is only present with -tunepme.

I ran a git bisection today to observe when the first commit occurred where the average potential energy of the four runs

```
mdrun -s topol.tpr -deffnm gpu-4-ranks -ntmpi 4 -gpu_id 0011 -v -tunepme
mdrun -s topol.tpr -deffnm gpu-2-ranks -ntmpi 2 -gpu_id 01 -v -tunepme
mdrun -s topol.tpr -deffnm gpu-2-ranks-sharing -ntmpi 2 -gpu_id 00 -v -tunepme
mdrun -s topol.tpr -deffnm gpu-1-ranks -ntmpi 1 -gpu_id 0 -v -tunepme
```
diverged from that of a CPU-only -ntmpi -1 run. The -ntmpi 1 seemed to sometimes agree with the CPU run (which was the basis of my previous offline comment to Szilard that there was no problem here; that the .tpr was made with gen-vel=yes was a red herring). Runs were done on tcbs23 (2 GPUs like Jan-Phillip) and amd1 (for the CPU-only).

Bisection showed that a8cf8c3 introduces the issue, but I cannot yet see why. CI regression testing wouldn't have detected this issue because we don't do any multi-GPU testing in Jenkins.

The average potential energies don't show any pattern with number of domains or degree of sharing, e.g. averaging over 40-400ps

```
cpu-1-ranks-a8cf8c3.log
Potential                -3.09989e+06         --       1479   -852.658  (kJ/mol)
gpu-1-ranks-a8cf8c3.log  Potential                -3.0841e+06         --    1778.82   -3075.32  (kJ/mol)
gpu-2-ranks-a8cf8c3.log  Potential                -3.05488e+06         --       1604   -2280.39  (kJ/mol)
gpu-2-ranks-sharing-a8cf8c3.log Potential                -3.08367e+06         --    1849.53   -3880.83  (kJ/mol)
gpu-4-ranks-a8cf8c3.log  Potential                -3.07667e+06         --    1821.81   -3642.99  (kJ/mol)
```

In the parent commit, I observe an order of magnitude smaller variation in average PE

```
cpu-1-ranks-15eb6d1.log
Potential                -3.0999e+06         --          0          0  (kJ/mol)
gpu-1-ranks-15eb6d1.log  Potential                -3.09783e+06         --          0          0  (kJ/mol)
gpu-2-ranks-15eb6d1.log  Potential                -3.09821e+06         --          0          0  (kJ/mol)
gpu-2-ranks-sharing-15eb6d1.log Potential                -3.10017e+06         --          0          0  (kJ/mol)
gpu-4-ranks-15eb6d1.log  Potential                -3.09781e+06         --          0          0  (kJ/mol)
```

These were only averages up to 40ps, but the above runs showed the above kind of variation by 40ps. I have longer runs going, but don't expect them to be relevant.

My suspicion is that the forces are unaffected, because the variation in each run over time tracks quite well between the different runs, but I don't have any serious evidence for that. The nature of the changes in the indicated commit suggests to me that some kernel that wasn't intended to be changed is no longer updating the short-range energy correctly because of an #ifdef mixup, but I can't see such an issue in the commit, nor how that relates only to runs with PME tuning. /confused
#4 - 09/25/2014 06:45 PM - Mark Abraham

There is a probably-unrelated bug at
http://redmine.gromacs.org/projects/gromacs/repository/revisions/a8cf8c39f389d529696015272a21cf24acc5bc15/entry/src/gromacs/mdlib/nbnxn_cuda/nbnxn_cuda.cu#L207

#5 - 09/26/2014 12:46 AM - Szilárd Páll

Thanks Mark for the detailed info! Obviously, there is something wrong with those energies, but having looked at the code I can't see what could be causing the problem; the switched LJ kernels have seemingly nothing to do with the twin cut-off code.

BTW, at the time when I last time had my hands on the CUDA jenkins setups, we did have a setup using AFAIR -ntpmi 2, hence two GPUs - will double check if it really has gone missing. What we don't have is tests with -tunepme.

#6 - 09/26/2014 12:47 AM - Szilárd Páll

Mark Abraham wrote:

There is a probably-unrelated bug at
http://redmine.gromacs.org/projects/gromacs/repository/revisions/a8cf8c39f389d529696015272a21cf24acc5bc15/entry/src/gromacs/mdlib/nbnxn_cuda/nbnxn_cuda.cu#L207

Indeed, that's a bug, but totally harmless (we've always had more electrostatics than LJ kernel types) and would only affect debug builds.

#7 - 09/26/2014 09:35 AM - Berk Hess

- Status changed from New to In Progress
- Priority changed from Normal to High

I found the issue.
In nbnxn_cuda_kernels.cuh VDW_CUTOFF_CHECK is replaced by LJ_CUTOFF_CHECK, but this has not been changed in nbnxn_cuda_kernel.cuh. This means that the LJ cut-off scales along with the Coulomb cut-off, which explains the effects. This is a serious bug, since it affects results, but could go unnoticed in many cases.

It would be nice if some static analysis can warn about unused and/or never set macros, but that's probably hard.

#8 - 09/26/2014 09:39 AM - Roland Schulz

I agree. Preprocessor usage is near to impossible for static analysis. Another good reason why we should replace preprocessor usage with some better alternative (e.g. templates).

#9 - 09/26/2014 12:01 PM - Szilárd Páll

- Assignee set to Szilárd Páll
- Target version changed from 5.x to 5.0.2
Berk Hess wrote:

I found the issue.
In nbnxn_cuda_kernels.cuh VDW_CUTOFF_CHECK is replaced by LJ_CUTOFF_CHECK, but this has not been changed in nbnxn_cuda_kernel.cuh. This means that the LJ cut-off scales along with the Coulomb cut-off, which explains the effects. This is a serious bug, since it affects results, but could go unnoticed in many cases.

Sadly, it looks indeed like a classic case of macro mixup. I'm not entirely sure why did I do the renaming, but it obviously did not end well.

What made it worse is that all regression tests (not just in jenkins) are run with with -notunepme.

#10 - 09/26/2014 12:07 PM - Gerrit Code Review Bot
Gerrit received a related patchset '1' for Issue #1603.
Uploader: Szilárd Páll (pall.szilard@gmail.com)
Change-Id: I43ae19968b30843cfe407e927a5c0fbd35c62881
Gerrit URL: https://gerrit.gromacs.org/4102

#11 - 09/26/2014 12:07 PM - Mark Abraham
Berk Hess wrote:

I found the issue.
In nbnxn_cuda_kernels.cuh VDW_CUTOFF_CHECK is replaced by LJ_CUTOFF_CHECK, but this has not been changed in nbnxn_cuda_kernel.cuh. This means that the LJ cut-off scales along with the Coulomb cut-off, which explains the effects. This is a serious bug, since it affects results, but could go unnoticed in many cases.

Average energies for LJ-short range on the parent commit:

```
[amd1 bisection ((15eb6d1...)|BISECTING)] $ ./do-analysis.sh cpu-1-ranks-15eb6d1.log
LJ (SR) 233970 -- 0 0 (kJ/mol)
gpu-1-ranks-15eb6d1.log
LJ (SR) 232593 -- 1464.95 210.98 (kJ/mol)
```

On the suspect commit:

```
[amd1 bisection ((15eb6d1...)|BISECTING)] $ git checkout a8cf8c3
Previous HEAD position was 15eb6d1... Improve floating point comparisons in unit tests
HEAD is now at a8cf8c3... CUDA kernels for switched LJ
[amd1 bisection ((a8cf8c3...)|BISECTING)] $ ./do-analysis.sh
cpu-1-ranks-a8cf8c3.log
LJ (SR) 232593 -- 1464.95 210.98 (kJ/mol)
```
So the theory that the LJ cutoff is getting changed inappropriately seems sound

#12 - 09/26/2014 12:20 PM - Mark Abraham

Szilárd Páll wrote:

Berk Hess wrote:

I found the issue.
In nbnxn_cuda_kernels.cuh VDW_CUTOFF_CHECK is replaced by LJ_CUTOFF_CHECK, but this has not been changed in nbnxn_cuda_kernel.cuh. This means that the LJ cut-off scales along with the Coulomb cut-off, which explains the effects. This is a serious bug, since it affects results, but could go unnoticed in many cases.

Sadly, it looks indeed like a classic case of macro mixup. I'm not entirely sure why did I do the renaming, but it obviously did not end well.

Well, we know a strategy that will help - do refactoring in one patch, and feature addition in another. It wouldn't have prevented this problem, but at least the bisection would show it was unrelated to the new kernels.

What made it worse is that all regression tests (not just in jenkins) are run with with -notunepme.

That's an unfortunate necessity if all we do is use gmx energy to do a component-wise diff of the energy files. In any case, none of them run enough steps for tuning to complete, so the difference might not even have been seen.

We need a real machine to dedicate to this style of per-commit testing (which will also be useful for performance regression testing) - particularly if we want to keep to the ~15-20 minute envelope for getting Jenkins feedback.

In May, while testing LJ-PME stuff, I wrote some machinery that we could use to observe that sums of energy components are what they should be, but the first patch in that tree (https://gerrit.gromacs.org/3475) sat around for 4 months before anybody other than Carsten reviewed it, so it's hard to see that we will ever be able to have nice things...
Mark Abraham wrote:

Szilárd Páll wrote:

Berk Hess wrote:

I found the issue.
In nbnxn_cuda_kernels.cuh VDW_CUTOFF_CHECK is replaced by LJ_CUTOFF_CHECK, but this has not been changed in nbnxn_cuda_kernel.cuh. This means that the LJ cut-off scales along with the Coulomb cut-off, which explains the effects. This is a serious bug, since it affects results, but could go unnoticed in many cases.

Sadly, it looks indeed like a classic case of macro mixup. I'm not entirely sure why did I do the renaming, but it obviously did not end well.

Well, we know a strategy that will help - do refactoring in one patch, and feature addition in another. It wouldn't have prevented this problem, but at least the bisection would show it was unrelated to the new kernels.

I agree, that would've been a good strategy as VDW_CUTOFF_CHECK had nothing to do with the feature introduced here. AFAIR the renaming had something to do with the way the variable/macro naming was changed in parent commits with CPU kernels, but that's irrelevant at this point.

What made it worse is that all regression tests (not just in jenkins) are run with with -notunepme.

That's an unfortunate necessity if all we do is use gmx energy to do a component-wise diff of the energy files.

That shouldn't have been hard to circumvent with a hack similar to some of the changes made in gmxtest since 4.6 first came out. The real problem IMO is that around the time of rel-4.6 the decision was made to not bother much with gmxtest because it'll be rewritten anyway. This approach has been applied since for about ~1.5-2 years and we've been leaving features, some introduced in 4.6, untested.

In any case, none of them run enough steps for tuning to complete, so the difference might not even have been seen.

Sure. One also needs reliable imbalance to test the load balancing. However, those are issues easy enough to solve - at least for those actively working on this code. As Berk pointed out offline today, we could have some built-in artificial imbalance triggered for testing - something like what DLB supports. For verification may also want to trigger a simplified single-pass balancing to make it faster, but still exercise the cut-off/grid changing and resetting. However, as the preferred and implemented solution was to use -notunepme there hasn't been a need for any of these solutions.

We need a real machine to dedicate to this style of per-commit testing (which will also be useful for performance regression testing) - particularly if we want to keep to the ~15-20 minute envelope for getting Jenkins feedback.

IMHO it's enough to do such tests nightly or weekly, and just ensure that testing of fragile parts of mdrun (like the PP-PME tuning) does happen instead of waiting for a more advanced testing framework to be implemented or new hardware (to be bought, installed, set up, etc.). Given that, based on your assessment, the average merge rate is 1/day, I don't think it's a big deal to dig through ~10 commit (or even 50) if one at least gets to know about an issue in time.

In May, while testing LJ-PME stuff, I wrote some machinery that we could use to observe that sums of energy components are what they should be, but the first patch in that tree ([https://gerrit.gromacs.org/3475](https://gerrit.gromacs.org/3475)) sat around for 4 months before anybody other than Carsten reviewed it, so it's hard to see that we will ever be able to have nice things...

We are getting quite off-topic here, let's discuss this further off-redmine.
By default, GPU-using PME-based mdrun simulations use a load-balancing scheme called "PME tuning" that changes the short-ranged Coulomb cutoff and Fourier grid spacing so that computational load is distributed as evenly as possible between the GPU and CPU, without perturbing the quality of the electrostatics part of the model physics. Unfortunately, an error was made in development such that instead of only changing the Coulomb cut-off, the VDW cut-off was moved along with it. This would tend to break these kinds of simulations by destroying the balance of the parameterized interactions within the force field.

Only simulations using mdrun from Gromacs 5.0 and 5.0.1 that used GPUs and PME tuning are affected - and PME tuning is on by default for such simulations. CPU-only simulations are unaffected. Simulations using LJ-PME and not electrostatics-PME are unaffected. Simulations using mdrun -notunepme are unaffected.

Affected runs can be observed by comparing the density of an NPT simulation with a correct run. Simulations in other ensembles are also affected, but a physical diagnostic is harder to observe. A log-file diagnostic can be found towards the end of the log file, for example:

PP - PME LOAD BALANCING

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial</th>
<th>Final</th>
<th>Cost-ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_cut Coulomb</td>
<td>1.000 nm</td>
<td>1.368 nm</td>
<td>2.35</td>
</tr>
<tr>
<td>r_cut VDW</td>
<td>1.118 nm</td>
<td>1.486 nm</td>
<td>0.33</td>
</tr>
</tbody>
</table>

If, for your simulation, the final r_cut value (1.368 here) is different from the initial one (1.000 here), then so was the VDW cutoff for short-ranged interactions, and the model physics was not what you asked for.

If the PME tuning decided not to shift any load to the GPU (e.g. because the GPU is relatively weak compared to the CPU) then you may observe that the initial and final values are the same. In that case, the earliest part of the simulation may have tried different r_cut values (examine for output after "Step 0"), which would have also changed rvdw. If you plan to discard the first part of the simulation as further equilibration, then we expect the system to re-equilibrate quickly from this small perturbation.

Simulations using vdwtype force-switch are affected, but less severely because of the shape of the force curve as it rises from zero after the intended cutoff distance.

Anything I missed? Overdid? Underdid?
I don't understand what you mean with: 
"Simulations using LJ-PME and not electrostatics-PME are unaffected."
Simulations with LJ-PME are unaffected, independently if you used PME for Coulomb (rcoulomb and rvdw scale) or not (no tuning at all then).

I was only aiming to cover the latter case (no tuning), but it sounds right that LJ-PME is always unaffected.

Revised with inclusion & replacement of stuff from Szilard's email. Any further thoughts?

The bug affects all GPU-accelerated PME simulations with 5.0 and 5.0.1 that were run with the default mdrun option -tunepme, and did actually change the electrostatics cut-off as a result of the PP-PME tuning. The bug means the effective LJ cutoff is the same as the tuned electrostatics cutoff, which will break most model physics.

CPU-only simulations are unaffected. Simulations using LJ-PME are unaffected. Simulations using mdrun -notunepme are unaffected.

Affected runs can be observed by comparing the density of an NPT simulation with a correct run. Simulations in other ensembles are also affected, generally showing a higher LJ potential energy. Your log file will show whether you are affected, because it reports the progress and result of the tuning. Near the end of the log file, you might see:

```
PP    PME LOAD BALANCING
PP/PM E load balancing changed the cut-off and PME settings:
          particle-particle   PME
            rcoulomb  rlist  grid  spacing  1/beta
  initial   1.000 nm  1.118 nm   120 120  96  0.115 nm  0.320 nm
  final    1.368 nm  1.486 nm   84  84  64  0.164 nm  0.438 nm
  cost-ratio  2.35        0.33
```

Note that these numbers concern only part of the total PP and PME load.

If, for your simulation, the final rcoulomb value (1.368 here) is different from the initial one (1.000 here), then so was the LJ cutoff for short-ranged interactions, and the model physics was not what you asked for.

If your log file is truncated somehow, you could grep your saved stderr output with

```bash
$ grep "timed with pme grid" your-stderr.txt
```

to see mdrun report on the tuning while it is in progress, and the optimum.

If the PME tuning decided not to shift any load to the GPU (e.g. because the GPU is relatively weak compared to the CPU) then you may observe that the initial and final rcoulomb values are the same. In that case, the earliest part of the simulation may have tried different rcoulomb values (examine for output after "Step 0"), which would have also changed rvdw. If you plan to discard the first part of the simulation as further equilibration, then we expect the system to re-equilibrate quickly from this small perturbation.
Simulations using vdwtype force-switch are affected, but less severely because of the shape of the force curve as it rises from zero after the intended cutoff distance.

#19 - 09/30/2014 06:18 PM - Szilárd Páll

Looks good.

On a second thought, itemizing the criteria listed in the 2nd and 3rd paragraphs for easier readability could help (but it's non-essential), e.g.

The bug affects simulations that:

- use versions 5.0 or 5.0.1 and
- use GPU-acceleration and
- use PME electrostatics and
- were run with the default mdrun option -tunepme, and did actually change the electrostatics cut-off as a result of the PP-PME tuning.

The bug means the effective LJ cutoff is the same as the tuned electrostatics cutoff, which will break most model physics.

The bug does not affect simulations that:

- use only CPUs or
- use LJ-PME or
- use mdrun -notunepme or did not change the electrostatics cut-off as a result of the PP-PME tuning.

#20 - 10/01/2014 02:23 PM - Mark Abraham

Release notes now live at [http://www.gromacs.org/About_Gromacs/Release_Notes/Versions_5.0.x](http://www.gromacs.org/About_Gromacs/Release_Notes/Versions_5.0.x), if any last changes

#21 - 10/01/2014 04:27 PM - Mark Abraham

- Status changed from Fix uploaded to Resolved

#22 - 10/01/2014 04:29 PM - Mark Abraham

- Status changed from Resolved to Closed

Files

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPUvsCPU.tar.gz</td>
<td>14.7 MB</td>
<td>09/22/2014</td>
<td>Jan-Philipp Machtens</td>
</tr>
</tbody>
</table>