### GROMACS - Bug #3100

**crash with GPU comm DD**

09/20/2019 02:36 AM - Szilárd Páll

<table>
<thead>
<tr>
<th>Status:</th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Category:</td>
<td>mdrun</td>
</tr>
<tr>
<td>Target version:</td>
<td>2020-beta2</td>
</tr>
<tr>
<td>Affected version:</td>
<td></td>
</tr>
<tr>
<td>Extra info:</td>
<td></td>
</tr>
<tr>
<td>Affected version:</td>
<td>git master</td>
</tr>
<tr>
<td>Difficulty:</td>
<td>uncategorized</td>
</tr>
</tbody>
</table>

**Description**

Fairly standard input, 4-rank PP-only offload run crashes after thousands of steps both with / without DLB and PME tuning. Input attached.

**Related issues:**

- Related to GROMACS - Feature #2890: GPU Halo Exchange **In Progress**

**Associated revisions**

Revision 2fb7901d - 09/20/2019 05:09 PM - Berk Hess

Fix DD 1 pulse checks without DLB

The check for requesting single pulse DD communication only worked correctly without DLB. Also added a check for 1 pulse with PME tuning.

Todo: PME tuned runs might still end up with multiple pulses due to pressure scaling. This needs to be checked.

Fixes #3100

Change-Id: lcac5e37ef79385fd7c7cf5c19c7b40b2d685b095b

**History**

#1 - 09/20/2019 02:40 AM - Szilárd Páll

- File topol.tpr added

#2 - 09/20/2019 10:03 AM - Alan Gray

Trying to reproduce.

@Mark, change 6fb2a76 Use GPU halo exchange only when compatible DD is available seems to have been over-zealous, and now this case (plus the cases I use for my development) are not actually trigerring GPU halo exchange (even with env variable set).

Rolling back to before this change, I have tested with gmx mdrun -s topol.tpr -ntomp 10 -pme cpu -nb gpu -ntmpi 4 -nsteps 100000 -v -notunepme -pin on -bonded gpu -dlb no -noconfout -gpu_id 0123 and it runs to completion.

@szilard, please can you let me know the mdrun command you used, which GPU you ran on and after how many steps did it crash.

Thanks

#3 - 09/20/2019 01:14 PM - Alan Gray

@Mark in is1DAnd1PulseDD() you are only returning true if (dd.comm->maxpulse == 1), but in these cases dd.comm->maxpulse=0 so it is returning false. Should it instead be (dd.comm->maxpulse <= 1)?

#4 - 09/20/2019 03:11 PM - Szilárd Páll

Alan Gray wrote:
@szilard, please can you let me know the mdrun command you used, which GPU you ran on and after how many steps did it crash.

```
GMX_GPU_PME_PP_COMMS=1 GMX_USE_GPU_BUFFER_OPS=1 GMX_GPU_DD_COMMS=1

gmx mdrun -nsteps 10000 -ntmpi 4
```

but it crashes with -dlb no as well as -notunepme. The run uses 4 ranks with 4x4 threads + 2 GPUs (so 2 ranks per GPU).

The amount of steps it takes is in the thousands (seen 2000-6000). This is a machine with a P6000 and a GTX TITAN, but I doubt that matters.

#5 - 09/20/2019 04:04 PM - Alan Gray

Thanks - now reproduced. (Noting that PME-PP comms are not relevant here and the bug is still present without GMX_GPU_PME_PP_COMMS=1)

In reference to previous similar race condition we fixed in GPU DD: we originally added a sync before both coordinate and force halo exchange, then we removed it for the force. This case runs successfully if the force sync is reinstated. Needs more work to understand why this is the case.

#6 - 09/20/2019 04:52 PM - Mark Abraham

Alan Gray wrote:

@Mark in is1DAnd1PulseDD() you are only returning true if (dd.comm->maxpulse == 1), but in these cases dd.comm->maxpulse=0 so it is returning false. Should it instead be (dd.comm->maxpulse <= 1)?

Berk's working on a fix for mdrun -dlb no at [https://gerrit.gromacs.org/c/gromacs/+/13283](https://gerrit.gromacs.org/c/gromacs/+/13283)

#7 - 09/20/2019 05:25 PM - Szilárd Páll

Alan Gray wrote:

(Noting that PME-PP comms are not relevant here and the bug is still present without GMX_GPU_PME_PP_COMMS=1)

Sure, I just triggered both in testing, but there was no PME offload so that didn't matter.

In reference to previous similar race condition we fixed in GPU DD: we originally added a sync before both coordinate and force halo exchange, then we removed it for the force.

Thanks for the update.

Are you referring to: [https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#318](https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#318) ?

I do not see why would an MPI sync call be necessary given that there is an explicit sync with the local stream ( [https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#266](https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#266) ) and implicit ordering on anything nonlocal in the nonlocal stream.

This case runs successfully if the force sync is reinstated. Needs more work to understand why this is the case.

I agree, we need a better understanding.

#8 - 09/20/2019 05:45 PM - Szilárd Páll

Szilárd Páll wrote:

Alan Gray wrote:

(Noting that PME-PP comms are not relevant here and the bug is still present without GMX_GPU_PME_PP_COMMS=1)

Sure, I just triggered both in testing, but there was no PME offload so that didn't matter.

In reference to previous similar race condition we fixed in GPU DD: we originally added a sync before both coordinate and force halo exchange, then we removed it for the force.

Thanks for the update.

Are you referring to: [https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#318](https://gerrit.gromacs.org/c/gromacs/+/12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#318) ?
I do not see why would an MPI sync call be necessary given that there is an explicit sync with the local stream (https://gerrit.gromacs.org/c/gromacs/+12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#266) and implicit ordering on anything nonlocal in the nonlocal stream.

...and just as I typed it in it has occurred to me: could it be a race with the bonded task that does write non-local forces, but we put that task in the nonlocal stream with DD so even that should not be an issue, I think (see forcerec.cpp:1988).

#9 - 09/20/2019 06:45 PM - Berk Hess
- Status changed from New to Resolved

Applied in changeset 2fb7901d11e4f7d21383898393bafae9b59dd96.

#10 - 09/23/2019 04:02 PM - Alan Gray

Are you referring to: https://gerrit.gromacs.org/c/gromacs/+12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#318?

Yes

I do not see why would an MPI sync call be necessary given that there is an explicit sync with the local stream (https://gerrit.gromacs.org/c/gromacs/+12943/16/src/gromacs/domdec/gpuhaloexchange_impl.cu#266) and implicit ordering on anything nonlocal in the nonlocal stream.

...and just as I typed it in it has occurred to me: could it be a race with the bonded task that does write non-local forces, but we put that task in the nonlocal stream with DD so even that should not be an issue, I think (see forcerec.cpp:1988).

Agreed - still looking into it.

#11 - 09/24/2019 10:14 AM - Alan Gray

could it be a race with the bonded task that does write non-local forces, but we put that task in the nonlocal stream with DD so even that should not be an issue, I think

Bonded forces will depend on both the local and non-local X buffer ops kernels, right? If bonded is in the non-local stream, then there should be a sync to ensure the local X buffer ops has completed, which I think may be missing in the code (or do we have that somewhere?). Not sure if that is the cause of this race, I guess its possible that use of the GPU halo exchange makes it more likely that the ordering goes wrong.

#12 - 09/24/2019 10:14 AM - Alan Gray
- Status changed from Resolved to In Progress

#13 - 09/24/2019 12:40 PM - Alan Gray

then there should be a sync to ensure the local X buffer ops has completed, which I think may be missing in the code (or do we have that somewhere?).

Actually, I think we already have this OK through the nbnxnInsertNonlocalGpuDependency() call at the end of the X buffer ops.

#14 - 09/24/2019 01:06 PM - Szilárd Páll

Alan Gray wrote:

then there should be a sync to ensure the local X buffer ops has completed, which I think may be missing in the code (or do we have that somewhere?).

Actually, I think we already have this OK through the nbnxnInsertNonlocalGpuDependency() call at the end of the X buffer ops.

Yes, that sync ensures all dependencies of non-local interactions on local atoms (coordinate transfer, and end-of-step including force buffer clearing and rolling pruning).

#15 - 09/24/2019 03:16 PM - Szilárd Páll
It's looking like the issue is with the conditional D2H copy of nonlocal coordinate data after the X halo exchange:

```c
if (domainWork.haveCpuBondedWork || domainWork.haveFreeEnergyWork)
{
    //non-local part of coordinate buffer must be copied back to host for CPU work
    nbv->launch_copy_x_from_gpu(as_rvec_array(x.unpaddedArrayRef().data()), Nbnxm::AtomLocality::NonLocal);
}
```

The condition is false for this case, but from debugging the issue it looks like it should be true. Can you think any extra condition we may require in the if statement?

#18 - 09/25/2019 10:46 PM - Mark Abraham
@Alan on what step? With nstlist of what?

#19 - 09/25/2019 11:46 PM - Alan Gray
I've looked at it some more and there is a possibility that the above extra copy caused the run to succeed not because of required data access on the host but instead through the side effect of delaying the nonlocal stream. Let me narrow it down some more and I'll report back.

#20 - 09/26/2019 11:15 AM - Alan Gray
I've found the bug. It's not in the GPU halo exchange, but was introduced in c69e061 Decouple coordinates buffer management from buffer ops in NBNXM (@artem)

The problem is that the new function to copy X from host to device for a given locality, when called in do_force to copy the local part, is actually copying extra atoms that encroach on the non-local part.

Originally, the number of local atoms was determined through gridSet.numRealAtomsLocal(). After the above change, the number of local atoms is instead determined through gridSet.grids()[0].atomIndexEnd(). These numbers don't match (at least for this test case), e.g. in one instance

```
gridSet.numRealAtomsLocal()=34085, gridSet.grids()[0].atomIndexEnd()=35328
```

Using the nbnxn_get_atom_range() function to get the atom range in the same way as the original code fixes the problem, as far as I can tell (I've run with and without the fix several times and get consistent results).

#21 - 09/26/2019 01:14 PM - Artem Zhmurov

Alan Gray wrote:

```
I've found the bug. It's not in the GPU halo exchange, but was introduced in c69e061 Decouple coordinates buffer management from buffer ops in NBNXM (@artem)

The problem is that the new function to copy X from host to device for a given locality, when called in do_force to copy the local part, is actually copying extra atoms that encroach on the non-local part.

Originally, the number of local atoms was determined through gridSet.numRealAtomsLocal(). After the above change, the number of local atoms is instead determined through gridSet.grids()[0].atomIndexEnd(). These numbers don't match (at least for this test case), e.g. in one instance

```

Using the nbnxn_get_atom_range() function to get the atom range in the same way as the original code fixes the problem, as far as I can tell (I've run with and without the fix several times and get consistent results).

Are you sure that nbnxn_get_atom_range(...) was used to get coordinate ranges? As far as I can tell it was only used for forces. The getAtomRanges(...) I've introduced is just an extraction of the code that was there into a separate function. I wanted to combine the two functions, but they had slightly different logic and my commit intended to be just refactoring. Anyhow, we need to handle these ranges properly. I think the class with AtomLocality enum and getAtomRanges(...) function should be the right solution. Quoting @Mark on this function: 

"Suggest also a todo that stuff like this should return a tuple (or a struct of first and size). For example

```c
int first, size;
std::tie(first, size) = getAtomRanges&lt;AtomLocality::All&gt;();
```

with
template<> std::tuple<int, int>
StatePropagatorDataGpu::Impl::getAtomRanges<AtomLocality::All>()
{
    return assertIfElementsAreNegative(std::make_tuple<int, int>(0, numAtomsAll_));
}

is easy to use, code, and document (only a template parameter to document, no pointers to assert are not null, no default switch case, no wondering if the compiler inlines away the switch branch). In turn, that motivates templating the copyTo/FromDevice functions on AtomLocality (suggest another todo).

As a quick fix though, reverting to just one correct getAtomRanges(...) function in NBNXM should do. I think.

#22 - 09/26/2019 02:06 PM - Alan Gray
Ok, yes the previous code was also using the same for the coordinates. I don’t understand why these are different - can someone explain? cr->dd->comm->atomRanges.numHomeAtoms() which is used in the halo exchange agrees with gridSet.numRealAtomsLocal().

#23 - 09/26/2019 02:31 PM - Artem Zhmurov
Alan Gray wrote:

Ok, yes the previous code was also using the same for the coordinates. I don’t understand why these are different - can someone explain? cr->dd->comm->atomRanges.numHomeAtoms() which is used in the halo exchange agrees with gridSet.numRealAtomsLocal().

As far as I can tell, it should be cr->dd->comm->atomRanges.numHomeAtoms() if there is no filler particles (which is the case right now). But @Berk is a much better source of knowledge here.

For now, I would use the right function for both coordinates and forces and remove the other function. It is not ideal, but it is a step forward.

#24 - 09/26/2019 03:33 PM - Alan Gray
Fix at https://gerrit.gromacs.org/c/gromacs/+/13401. @Szilard, can you just check that this also fixes your run?

#25 - 09/27/2019 01:27 PM - Szilárd Páll
Alan Gray wrote:

Fix at https://gerrit.gromacs.org/c/gromacs/+/13401. @Szilard, can you just check that this also fixes your run?

Yes, it looks like it does. However, as noted on gerrit, so does https://gerrit.gromacs.org/c/gromacs/+/12919 which is preferred as it eliminates a lot of the technical a debt introduced earlier.

#26 - 09/27/2019 01:28 PM - Szilárd Páll
Szilárd Páll wrote:

Alan Gray wrote:

Fix at https://gerrit.gromacs.org/c/gromacs/+/13401. @Szilard, can you just check that this also fixes your run?

Yes, it looks like it does. However, as noted on gerrit, so does https://gerrit.gromacs.org/c/gromacs/+/12919 which is preferred as it eliminates a lot of the technical a debt introduced earlier.

side-note: 12919 does eliminate this issue as well in tests, but as it requires excessive sync-ing the lack of crashes is a less certain indicator.

#27 - 11/01/2019 11:12 AM - Alan Gray
- Status changed from In Progress to Closed

Files
topol.tpr 7.94 MB 09/20/2019 Szilárd Páll