GROMACS - Bug #2360

error at counter reset with PME-only rank

12/21/2017 03:17 AM - Szilárd Páll

Status:	New		
Priority:	Low		
Assignee:			
Category:	mdrun		
Target version:	future		
Affected version - extra info:		**Difficulty:**	hard
Affected version:	2016.5		

Description

Counter resetting fails with a separate PME rank. It seems that the internal state of the PME load balancer is not updated correctly and the balancing never reaches a stopped state.

```
mdrun -ntmpi 4 -ntomp 8 -noconfout -pin on -npme 1 -nsteps 7000 -resetstep 5000 -nb gpu -pme gpu -
v
[...]
Using 4 MPI threads
Using 8 OpenMP threads per tMPI thread

On host threadripper-gpu01 1 GPU auto-selected for this run.
Mapping of GPU IDs to the 4 GPU tasks in the 4 ranks on this node:
  PP:0,PP:0,PP:0,PME:0

NOTE: DLB will not turn on during the first phase of PME tuning
starting mdrun 'Water'
7000 steps,     14.0 ps.
step  320: timed with pme grid 112 112 112, coulomb cutoff 0.900: 711.5 M-cycles
step  480: timed with pme grid 100 100 100, coulomb cutoff 0.997: 748.6 M-cycles
step  640: timed with pme grid 84 84 84, coulomb cutoff 1.187: 853.4 M-cycles
step  800: timed with pme grid 96 96 96, coulomb cutoff 1.038: 740.4 M-cycles
step  960: timed with pme grid 100 100 100, coulomb cutoff 0.997: 728.9 M-cycles
step 1120: timed with pme grid 104 104 104, coulomb cutoff 0.958: 805.1 M-cycles
step 1280: timed with pme grid 108 108 108, coulomb cutoff 0.923: 710.2 M-cycles
step 1440: timed with pme grid 112 112 112, coulomb cutoff 0.900: 701.8 M-cycles
step 1600: timed with pme grid 96 96 96, coulomb cutoff 1.038: 723.0 M-cycles
step 1760: timed with pme grid 100 100 100, coulomb cutoff 0.997: 775.2 M-cycles
step 1920: timed with pme grid 104 104 104, coulomb cutoff 0.958: 818.0 M-cycles
step 2080: timed with pme grid 108 108 108, coulomb cutoff 0.923: 704.7 M-cycles
step 2240: timed with pme grid 112 112 112, coulomb cutoff 0.900: 711.5 M-cycles
step 2400: timed with pme grid 96 96 96, coulomb cutoff 1.038: 722.0 M-cycles
step 2560: timed with pme grid 100 100 100, coulomb cutoff 0.997: 725.4 M-cycles
step 2720: timed with pme grid 108 108 108, coulomb cutoff 0.923: 713.2 M-cycles
step 2880: timed with pme grid 112 112 112, coulomb cutoff 0.900: 695.5 M-cycles
              optimal pme grid 112 112 112, coulomb cutoff 0.900

NOTE: DLB can now turn on, when beneficial
step 4900, remaining wall clock time:     9 s          imb F  0% pme/F 1.14
-------------------------------------------------------
Program:     gmx mdrun, version 2018-beta2-dev-20171211-2e91fcf-dirty
Source file: src/programs/mdrun/md.cpp (line 1930)
MPI rank:    2 (out of 4)

Fatal error:
PME tuning was still active when attempting to reset mdrun counters at step
5000. Try resetting counters later in the run, e.g. with gmx mdrun -resetstep.

For more information and tips for troubleshooting, please check the GROMACS
website at http://www.gromacs.org/Documentation/Errors
```

_____

Related issues:

Related to GROMACS - Task #1781: re-design benchmarking functionality	**Accepted**

History

#1 - 12/21/2017 03:17 AM - Szilárd Páll

- Affected version changed from 2016.3 to 2018-beta3

#2 - 12/21/2017 03:20 AM - Szilárd Páll

AFAICT there are subtle issue already at the pme_loadbal_init() call (bUseGPU) and the way bActive/bBalance is set and later changed.

#3 - 12/21/2017 03:11 PM - Mark Abraham

We should disable tuning for this case, since no planning has gone into it.

#4 - 12/21/2017 05:52 PM - Aleksei Iupinov

I get the same with -pme cpu

#5 - 01/03/2018 06:48 PM - Mark Abraham

- Related to Task #1781: re-design benchmarking functionality added

#6 - 01/03/2018 06:49 PM - Mark Abraham

Aleksei is looking into it.

I also propose we remove these reset features in master branch until someone puts in enough thought to have it work stably and be testable.

#7 - 01/03/2018 06:50 PM - Mark Abraham

- Target version set to 2018

#8 - 01/03/2018 08:20 PM - Szilárd Páll

Mark Abraham wrote:

> I also propose we remove these reset features in master branch until someone puts in enough thought to have it work stably and be testable.

The reset feature is not the issue, but rather the complex and fragile internal state management of the PME tuner.
Not sure how does removing the resetting feature accomplish anything useful, if anything it will make performance measurement impossible (or at least unreasonably hard).

#9 - 01/04/2018 09:57 AM - Erik Lindahl

I have no idea whether this is a special fragility with the PME tuner, but there are definitely many fragile parts of the code. However, there is no "other" part of the team whose job it is to make everything non-fragile, so if somebody wants/needs to keep a particular feature that exposes other fragility, I expect they will volunteer and make it a priority to help fix those fragilities with properly designed and documented classes.

#10 - 01/04/2018 11:26 AM - Mark Abraham

Szilárd Páll wrote:

> Mark Abraham wrote:
>
> > I also propose we remove these reset features in master branch until someone puts in enough thought to have it work stably and be testable.
>
> The reset feature is not the issue, but rather the complex and fragile internal state management of the PME tuner.

Indeed.

> Not sure how does removing the resetting feature accomplish anything useful, if anything it will make performance measurement impossible (or at least unreasonably hard).

I proposed back in June at #1781 removing -nsteps, multiple people objected, and multiple people have developed their performance features instead of working on the performance measurement infrastructure. That Redmine has been open since July 2015. That's sounding like people are voting with their feet for not having such infrastructure. Even if we do agree to actually work on this, it's not going to take the form of -resetstep, so I don't see

the value in retaining it. Implementing mdrun -benchmarksteps 5000 (or whatever) is not going to be easier if -resetstep and friends are still around.

#11 - 01/04/2018 12:27 PM - Aleksei Iupinov

- Subject changed from error at counter reset with PME-only rank + GPU to error at counter reset with PME-only rank

- Description updated

- Priority changed from Normal to Low

- Target version changed from 2018 to future

- Affected version changed from 2018-beta3 to 2016.5

#12 - 01/04/2018 12:27 PM - Aleksei Iupinov

This has to do with balancing (pme_loadbal_do) in case of separate PME rank(s), not specifically PME GPU/CPU. The code is old, so same happens in release-2016, too.

What is happening is that at optimal grid settings pme_lb->bTriggerOnDLB is set to TRUE and the "NOTE: DLB can now turn on, when beneficial" message is printed.
For the balancing to be retriggered on DLB imbalance, another grace period of PMETunePeriod(= 50) * nstlist(= 80 in your case?) steps is given.
After that period passes without balancing being retriggered, balancing is actually switched off forever.
So if you'd added a couple more thousand steps to nsteps/resetstep, it should have worked.
I would argue the message is correct in this case, but the program is not clearly communicating the additional bTriggerOnDLB period.

What is slightly concerning is that in case of non-separate PME ranks the whole bTriggerOnDLB is set but not actually used (balancing deactivation happens same step anyway), so that is why you can reset counters sooner without separate PME ranks. This is done by this code (pme-load-balancing.cpp:1059):

```
    if (!pme_lb->bBalance &&
        (!pme_lb->bSepPMERanks || step_rel > pme_lb->step_rel_stop))
    {
        /* We have just deactivated the balancing and we're not measuring PP/PME
         * imbalance during the first steps of the run: deactivate the tuning.
         */
        pme_lb->bActive = FALSE;
    }
```

Judging from the rest of the code, balancing on the DLB is only intended for separate PME ranks - there are pme_lb->bSepPMERanks checks all over the place.
So I would say this is not a bug, but reworking all those booleans into a reasonable enum would be great in master.
We could also consider additional "balancing switched off forever" message, but seeing how this only got uncovered by using a benchmarking feature, this is probably not worth spamming regular users.

#13 - 01/04/2018 02:11 PM - Gerrit Code Review Bot

Gerrit received a related patchset '2' for Issue #2360.
Uploader: Aleksei Iupinov (a.yupinov@gmail.com)
Change-Id: gromacs~master~I6d32b845fc19a0b0b328cbd3ed96e1cbb876b07a
Gerrit URL: https://gerrit.gromacs.org/7426

#14 - 01/04/2018 02:38 PM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2360.
Uploader: Aleksei Iupinov (a.yupinov@gmail.com)
Change-Id: gromacs~master~Ifbe787739ad3d52cc1df107fa05f692c192dcb5c
Gerrit URL: https://gerrit.gromacs.org/7427

#15 - 01/04/2018 02:53 PM - Mark Abraham

Concur that there's no user-facing behaviour that needs a fix in any release branch.

#16 - 01/04/2018 03:57 PM - Szilárd Páll

Aleksei Iupinov wrote:

> This has to do with balancing (pme_loadbal_do) in case of separate PME rank(s), not specifically PME GPU/CPU. The code is old, so same happens in release-2016, too.

Thanks for checking. Makes sense now, given the larger nslist this is more likely to occur.

> What is happening is that at optimal grid settings pme_lb->bTriggerOnDLB is set to TRUE and the "NOTE: DLB can now turn on, when beneficial" message is printed.
> For the balancing to be retriggered on DLB imbalance, another grace period of PMETunePeriod(= 50) * nstlist(= 80 in your case?) steps is given.
> After that period passes without balancing being retriggered, balancing is actually switched off forever.

> So if you'd added a couple more thousand steps to nsteps/resetstep, it should have worked.
> I would argue the message is correct in this case, but the program is not clearly communicating the additional bTriggerOnDLB period.
>
> What is slightly concerning is that in case of non-separate PME ranks the whole bTriggerOnDLB is set but not actually used (balancing deactivation happens same step anyway)

That is not great code-wise, but seems fine from a correctness pov as AFAICT the implementation assumes that it should only re-balance after DLB is turned on when there is the PP-PME imbalance is measurable (more docs/assertions on this would be helpful).

> Judging from the rest of the code, balancing on the DLB is only intended for separate PME ranks - there are pme_lb->bSepPMERanks checks all over the place.

Yes, because the assumption is that that's the only case where imbalance can be measured (and this is valid given that, despite it being possible, we have not implemented such measurements with OpenCL).

> So I would say this is not a bug, but reworking all those booleans into a reasonable enum would be great in master.
> We could also consider additional "balancing switched off forever" message, but seeing how this only got uncovered by using a benchmarking feature, this is probably not worth spamming regular users.

The error message could also be improved if we'd query the actual state of the balancer and issued a slightly different message.

#17 - 01/04/2018 04:35 PM - Szilárd Páll

Erik Lindahl wrote:

> I have no idea whether this is a special fragility with the PME tuner, but there are definitely many fragile parts of the code. However, there is no "other" part of the team whose job it is to make everything non-fragile, so if somebody wants/needs to keep a particular feature that exposes other fragility,

A feature triggers some other issue/bug/fragility so the said feature should be removed? Sounds backward, we should be suggesting removal of
Did you meant he other way around?

Also unsure who here suggested that some "other" person should fix something.

> I expect they will volunteer and make it a priority to help fix those fragilities with properly designed and documented classes.

If there are suggestions to improve the load balancer, can we discuss that directly instead?

Also, can we stick to technical details that relate to this issue rather than bringing up anything remotely related that might seem to motivate some feature removal, the lack of "proper C++", etc.?

#18 - 01/04/2018 04:41 PM - Szilárd Páll

Aleksei, while looking at this yesterday, it seemed to me that the code never switched the tuning to bActive=false (while r2016 seemed to do the switch), but admittedly it could have been because I was running a debug build.

#19 - 01/04/2018 05:28 PM - Erik Lindahl

Szilard, since you asked: Nobody is happier than me if we get it working well in the release, but remember the context of the discussion: This was something that appeared to be a non-trivial bug that did not have any assignee, there were no comments added that indicated anybody had started to work on for two weeks after it appeared, and then the opinion came up that it should be a release blocker.

I too like some features, but we also have to accept that all other code is not perfect, so there will be cases where we might have to (maybe temporarily) disable a feature simply because that is trivial to do, while fixing other heavily used code that causes problems in combination with this feature would take too much work. That is not a vote for some code being better than other, but that we need to have a working release.

#20 - 01/05/2018 12:43 PM - Mark Abraham

Unless there's a user-facing issue, there's no urgency to do anything about this for 2018 release. We have lots of infrastructure needing major improvement, and we need to be on the ruthless side of realistic that we should put our effort on moving toward better future rather than addressing the past.