GROMACS - Feature #3142

Feature # 2816 (Closed): GPU offload / optimization for update&constraits, buffer ops and multi-gpu communication

Feature # 2817 (Closed): GPU X/F buffer ops

Feature # 3029 (Closed): GPU force buffer ops + reduction

centralize and clarify GPU force buffer clearing

10/14/2019 04:53 PM - Szilárd Páll

Status:	Closed
Priority:	High
Assignee:	
Category:	mdrun
Target version:	2021-infrastructure-stable
Difficulty:	uncategorized

Description

The responsibility of (rvec) force buffer clearing should be moved into StatePropagatorDataGpu and arranged for such that this is not a task on the critical path (as it as right now in GpuHaloExchange::Impl::communicateHaloForces()).

At the same time, we need to

- skip CPU-side force buffer clearing if there are no CPU forces computed
- check all code-paths and make sure we can not end up with reduction kernels accumulating into non-initialized buffers.

History

#1 - 10/14/2019 04:56 PM - Szilárd Páll

Note: this is high priority, but we can consider bumping it to beta3 **except** the last item and its impact on the buffer ops +/- GPU update code-path.

#2 - 10/25/2019 12:15 PM - Alan Gray

> The responsibility of (rvec) force buffer clearing should be moved into StatePropagatorDataGpu and arranged for such that this is not a task on the critical path (as it as right now in GpuHaloExchange::Impl::communicateHaloForces()).

This has been done in the pending change
https://gerrit.gromacs.org/c/gromacs/+/13885

It is now a StatePropagatorDataGpu method operating in the non local stream.

#3 - 11/01/2019 03:21 PM - Paul Bauer

- Target version changed from 2020-beta2 to 2020-beta3

bump

#4 - 11/01/2019 04:28 PM - Szilárd Páll

- Status changed from New to In Progress

#5 - 11/27/2019 05:28 PM - Szilárd Páll

Szilárd Páll wrote:

> At the same time, we need to
>
> - skip CPU-side force buffer clearing if there are no CPU forces computed

Quick test shows that switching off force clearing saves 7% runtime (with 6k/core); as this is typically offloaded, it only contributes to avoiding cache pollution. However, switching to doing force reduction (when this is done on the CPU) to store instead of accumulate saves another 6-8% of runtime (assuming that the run is completely GPU bound), so that is in the tested case a potential ~15% performance improvement when there are no CPU force tasks but reduction is done on the CPU.

This would especially help the default GPU code-path (reductions on CPU with CUDA as well as the OpenCL path).

- check all code-paths and make sure we can not end up with reduction kernels accumulating into non-initialized buffers.

Mostly done. An issue going forward identified, filed as a separate redmine (#3216).

#6 - 11/27/2019 08:13 PM - Artem Zhmurov

Shall I revive https://gerrit.gromacs.org/#/c/gromacs/+/13358/ then?

#7 - 12/02/2019 09:52 AM - Paul Bauer

- Target version changed from 2020-beta3 to 2021-infrastructure-stable

2021 is a more realistic target

#8 - 12/02/2019 12:09 PM - Szilárd Páll

What Artem suggested is a task/cleanup related to the GPU comm feature, so indeed that should have been done by now

The rest are WIP (in particular the conditional clearing/reduction) and should be addressed IMO. It would be a great disservice to our users to not allow getting the benefits of *existing* code (especially if we instead focus on adding more unstable code).

#9 - 02/06/2020 11:49 AM - Alan Gray

- Status changed from In Progress to Closed

Moved to umbrella task https://redmine.gromacs.org/issues/3370