GROMACS - Task #3370

Further improvements to GPU Buffer Ops and Comms

02/06/2020 11:11 AM - Alan Gray

Status:	New
Priority:	High
Assignee:	
Category:	
Target version:	
Difficulty:	uncategorized

Description

Umbrella task for follow-up improvements.

[HIGH PRIORITY] Unification of code-paths across different types of step in do_force

- Allow GPU Force Buffer ops to be active on virial steps
  - Fix uploaded https://gerrit.gromacs.org/c/gromacs/+/15960
- ~~Unify X/F Buffer ops flags~~
  - ~~Above fix unifies to single stepWork flag~~
  - https://gerrit.gromacs.org/c/gromacs/+/15961 ~~moves to unified simulationWork flag~~
  - see comments below
- Allow GPU PME-PP comms to be active on virial steps
- Allow GPU halo exchange to be active on virial steps (requires extension to include shift force contribution)
- ~~Unify and simplify X/F Halo exchange triggers.~~ See comments below.
- ~~Allow GPU X buffer ops to be active on search steps. Update: realized this is not required since there are no X buffer ops calls from do_force on search steps.~~

[HIGH PRIORITY] Refactoring

- ~~eliminate regression due to moving gmx_pme_send_coordinates()~~
  - ~~subtask~~ https://redmine.gromacs.org/issues/3159
  - Fix uploaded https://gerrit.gromacs.org/c/gromacs/+/15200
- move ddUsesGpuDirectCommunication and related conditionals into the workload data structures
  - Fix uploaded https://gerrit.gromacs.org/c/gromacs/+/15437

[HIGH PRIORITY] Force buffer op and reduction cleanup/improvement

Previous general discussion at https://redmine.gromacs.org/issues/3029

- Rework GPU direct halo-exchange related force reduction complexities
  - subtask https://redmine.gromacs.org/issues/3093
- Centralize and clarify GPU force buffer clearing: 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()).
  - Previous discussion at https://redmine.gromacs.org/issues/3142
- 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.
- Launch the transform kernel back-to-back after the nonbonded rather than later, next to the CPU buffer ops/reduction
- the transform+reduce kernels can use simple or atomic accumulation into a reduced f output buffer; the former will require exclusive access to the target force buffer (need to wait for the completion of any kernel that produces forces into it) while the latter would only require a wait on the source force buffer(s) to be reduced into the target (e.g. GPU NB and/or CPU force buffer).
- consider inline transform function for on-the-fly transform within the nonbonded kernel; in particular for high parallelization the performance hit in the nonbonded kernel may be less than the cost of launching an extra kernel.
- Ideally the force-reduction should not be called from a method of the nonbonded module (especially due to the complexities of CPU/GPU code-paths) - consider reorganizing reductions

Remove Limitations

- ~~Implement multiple pulses within GPU halo exchange communication~~
  - ~~subtask~~ https://redmine.gromacs.org/issues/3106
  - Fix uploaded https://gerrit.gromacs.org/c/gromacs/+/14723
- Implement multiple dimensions within GPU halo exchange communication
  - Fix uploaded https://gerrit.gromacs.org/c/gromacs/+/16181
- Extend PME-PP communication to support case where PME is on CPU and PP is on GPU.
  - subtask https://redmine.gromacs.org/issues/3160
  - fix uploaded https://gerrit.gromacs.org/c/gromacs/+/14223
- Extend PME-PP communication to support coordinate send from CPU
  - subtask https://redmine.gromacs.org/issues/3160
  - fix uploaded https://gerrit.gromacs.org/c/gromacs/+/14238

Timing

- add missing cycle counters related to buffer ops/reduction launches

Improve synchronization

- Implement better receiver ready / notify in halo exchange: Current notification mechanisms render the one-sided communication synchronous two-sided. Alternatives should be considered.
- Separate PME x receive sync: the data dependency sychronization should be implemented on the consumer task's end which is PME spread in the case of PME. PME-only ranks have the receive enqueue wait as soon as MPI returns. Consider assembling a list of events and passed to spread instead. Consider whether having to receive from multiple PP ranks actually makes is more beneficial to overlap some receive with event wait enqueue.

Investigate GPU f buffer ops use cases

Check if there is any performance benefits to be had and in which regimes for x / f buffer opts without GPU update in:

- runs with DD and CPU update
  - x buffer ops: offloadable with a likely simple crossover heuristic threshold; i.e. below N atoms/core not offloaded (locals or also nonlocals, with/without CPU work?)
  - f buffer ops: heuristics likely more complex criteria (as it is combined with reductions)
- runs with / without DD and vsites
  - with GPU update requires D2H and H2D -- is it worth it, test use-cases (e.g. multiple ranks per GPU, both ensemble and DD runs, transfers might be overlapped)
  - without GPU update: same applies as above non-vistes runs just wait on D2H needs to be earlier

evaluate what is #atoms threshold under which it is not worth taking the 10-15 us overhead of kernel launch (especially for non-local buffer ops)

Subtasks:

Feature # 2890: GPU Halo Exchange	**In Progress**
Task # 3089: relax dlb scaling limit when that would suit GPU halo exchange	**Closed**
Task # 3092: implement better receiver ready / notify in halo exchange	**Closed**
Task # 3104: implement GPU DD cycle counting	**New**
Task # 3156: move ddUsesGpuDirectCommunication and related conditionals into the worklo...	**Closed**
Feature # 2891: PME/PP GPU communications	**In Progress**
Task # 3077: PME/PP GPU Comms unique pointer deletion causes seg fault when CUDA calls ...	**Feedback wanted**
Task # 3105: implement GPU PME/PP comm cycle counting	**New**
Task # 3157: separate PME x receive sync	**Closed**
Task # 3158: use MPI_Isend() in sendFToPpCudaDirect and receiveCoordinatesFromPpCudaDirect	**Closed**
Bug # 3164: mdrun-mpi-test with separate PME ranks and PP-PME CPU comm crashes	**Closed**
Feature # 2915: GPU direct communications	**In Progress**
Task # 3082: move launch/synchronization points to clarify task dependencies	**New**
Feature # 3087: enable GPU peer to peer access	**Closed**
Task # 2965: Performance of GPU direct communications	**In Progress**
Feature # 3021: Completion of docs for GPU developments	**Feedback wanted**
Task # 3093: rework GPU direct halo-exchange related force reduction complexities	**In Progress**
Task # 3106: Implement multiple pulses with GPU communication	**Closed**
Bug # 3159: eliminate regression due to moving gmx_pme_send_coordinates()	**Closed**

Associated revisions

Revision e6650c00 - 09/28/2020 10:53 PM - Alan Gray

Remove single-dimension limitation from GPU halo exchange

Allows GPU halo exchange to be active when the number of dimensions is greater than 1, thus simplifying the codepath logic.

Partly addresses #3370

Revision 257d8094 - 09/29/2020 01:27 PM - Alan Gray

Redevelopment of GPU Force Reduction/Buffer Ops

Introduces a new purpose-build class for GPU force reduction, which replaces the previous force buffer ops mechanism.

Refs #3370

Revision ed556fe1 - 10/02/2020 09:02 AM - Alan Gray

Use workload data structures for GPU halo exchange triggers

Move GPU halo exchange trigger booleans and related conditionals into workload data structures, and remove unnecessary assertion on GPU buffer ops being active (since it is now automatically activated when GPU halo exchange is active).

Partly addresses #3370

History

#1 - 02/06/2020 11:17 AM - Alan Gray

- Description updated

#2 - 02/06/2020 11:18 AM - Alan Gray

- Description updated

#3 - 02/06/2020 11:22 AM - Alan Gray

- Description updated

#4 - 02/06/2020 11:26 AM - Alan Gray

- Description updated

#5 - 02/06/2020 11:31 AM - Alan Gray

- Description updated

#6 - 02/06/2020 11:46 AM - Alan Gray

- Description updated

#7 - 02/06/2020 11:50 AM - Alan Gray

- Description updated

#8 - 02/06/2020 11:55 AM - Alan Gray

- Description updated

#9 - 02/06/2020 11:56 AM - Alan Gray

- Description updated

#10 - 02/07/2020 01:03 PM - Alan Gray

- Description updated

#11 - 02/07/2020 01:05 PM - Alan Gray

- Description updated

#12 - 02/07/2020 01:26 PM - Alan Gray

- Description updated

#13 - 02/14/2020 12:00 PM - Alan Gray

- Description updated

#14 - 02/16/2020 09:14 PM - Alan Gray

- Description updated

#15 - 02/20/2020 11:59 AM - Alan Gray

- Description updated

#16 - 02/20/2020 08:15 PM - Szilárd Páll

- Unify X/F Buffer ops flags
- x buffer ops: offloadable with a likely simple crossover heuristic threshold; i.e. below N atoms/core not offloaded (locals or also nonlocals, with/without CPU work?)

Have we done measurements of the cross-over of CPU time of CPU vs GPU buffer ops? If we have not, the above goals do conflict. Unifying the flags means the x buffer ops trigger can not be tuned based on a #atoms threshold.

Additionally, the x/f buffer ops are entirely different tasks so I see little benefit in merging their workload flags -- other than saving a few bytes in the workload data structure.

- ~~Allow GPU X buffer ops to be active on search steps. Update: realized this is not required since there are no X buffer ops calls from do_force on search steps.~~

On search steps the search produces nonbonded-layout x, so technically it is not needed. We could change that and avoid having the search store the coordinates and call the buffer ops instead. The benefit would be uniform behavior on the GPU across all steps *but* different behavior for CPU and GPU search.

#17 - 02/20/2020 08:20 PM - Szilárd Páll

Szilárd Páll wrote:

> - Unify X/F Buffer ops flags
> - x buffer ops: offloadable with a likely simple crossover heuristic threshold; i.e. below N atoms/core not offloaded (locals or also nonlocals, with/without CPU work?)
>
> Have we done measurements of the cross-over of CPU time of CPU vs GPU buffer ops?

See https://redmine.gromacs.org/issues/3029#note-1; IIRC that was CPU vs GPU kernel time, but CPU critical path will be affected more by kernel launch cost (at least until we can overlap GPU launch with CPU execution).

#18 - 02/21/2020 09:25 AM - Alan Gray

- Description updated

#19 - 02/21/2020 10:04 AM - Alan Gray

The idea is to try and simplify the logic in do_force (and do_md) by unifying flags and ultimately code-paths, to improve readability and maintainability of the code (and reduce the scope of required test coverage). I acknowledge that this is in conflict with the idea of developments that add more flexibility to hardware scheduling through heuristics, which may have some performance benefits but would further increase complexity. I suggest that we focus on simplification/cleanup in the short term, and put the latter idea on the backburner as a possible future optimization task.

#20 - 02/21/2020 12:04 PM - Artem Zhmurov

Alan Gray wrote:

> The idea is to try and simplify the logic in do_force (and do_md) by unifying flags and ultimately code-paths, to improve readability and maintainability of the code (and reduce the scope of required test coverage). I acknowledge that this is in conflict with the idea of developments that add more flexibility to hardware scheduling through heuristics, which may have some performance benefits but would further increase complexity. I suggest that we focus on simplification/cleanup in the short term, and put the latter idea on the backburner as a possible future optimization task.

I agree with Szilard here. The X buffer ops most likely should be enabled when we need coordinates in nbat format on the GPU. I think we can even live with re-doing them on the GPU on search steps, thus the XBuffOps flag will naturally go away. This will also eliminate the need of copy_nbat_coordinates_host_to_device function and logic around it. F buffer ops need more work before the corresponding flag is eliminated the same way.

#21 - 02/21/2020 01:57 PM - Szilárd Páll

This is not just a matter of leaving room for optimization. The two tasks in question. x buffer ops and f buffer ops + reduction, are entirely different tasks, so it does make sense to keep them separate.

Also note that tasks themselves will not be "eliminated" (unless underlying algorithms change) and therefore it is entirely reasonable to have workload flags corresponding to these. These flags define the schedule and therefore, unless a task becomes trivial or merged into another across all code paths (i.e. all GPU code-paths support X buffer ops and always schedule it together with some other nbnxm task), the XBuffOps flag can't go away.

Last, I see only a very small code (LOC/logic) siomplification in using one simulationWorkload.useGpuBufferOps versus two stepWorkload workload flags.
Sode-note: the current thinking is that we should have an inclusive stepWorkload data structure that contains all the higher level flags and is constructed ahead of time for N steps.

#22 - 02/21/2020 02:02 PM - Alan Gray

OK, noted. I will update the existing patches in gerrit accordingly.

#23 - 02/21/2020 05:57 PM - Alan Gray

- Description updated

#24 - 03/04/2020 01:38 PM - Alan Gray

- Description updated