GROMACS - Feature #2817

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

GPU X/F buffer ops

12/21/2018 05:50 PM - Szilárd Páll

Status:	In Progress
Priority:	High
Assignee:	
Category:	mdrun
Target version:	2020-beta3
Difficulty:	uncategorized

Description

Implement the native to nbat/nbnxn layout transforms/"buffer ops" in the nbnxn_gpu module (nbnxn_atomdata_add_nbat_f_to_f and nbnxn_atomdata_copy_x_to_nbat_x).

Role and scope:

- Implementing the coordinate X transform on the GPU will allow transferring only the native layout. While this may not make the code faster -- considering the CUDA API overheads and that the "extra" H2D transfer is typically overlapped --, it does remove the CPU form the critical path in nonbonded communication which is beneficial to scaling and direct GPU-GPU communication.
- Similarly, the force layout transform kernel will allow direct force communication. This transform can be combined with reduction across force buffers. Multiple flavors and implementation strategies to be considered:
  - only transform (e.g. if no other force compute on the GPU)
  - transform + reduce by accumulating into the f buffer output on the GPU (e.g. reduce the NB and the RVec f PME GPU output);
  - transform + reduce multiple force buffers: reduce the result of force calculation outputs from different memory spaces (special forces on CPU, PME on separate GPU, etc.)
  - 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.

Related TODOs:

- need to improve resolve ownership of GPU input/outputs [WIP]
- ~~pinning for currently not pinned/pinnable search data~~
- 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

Subtasks:

Feature # 2934: GPU X Buffer ops	**New**
Task # 3026: add flags for GPU force buffer op / reduction activation	**New**
Feature # 3029: GPU force buffer ops + reduction	**In Progress**
Feature # 3052: GPU virial reduction/calculation	**New**
Task # 3128: do not fall back to CPU path on energy-only steps	**Closed**
Feature # 3142: centralize and clarify GPU force buffer clearing	**In Progress**
Task # 3170: investigate GPU f buffer ops use cases	**New**
Task # 3037: add missing cylcle counters related to buffer ops/reduction launches	**New**

Related issues:

Related to GROMACS - Task #3171: schedule CPU H2D force contribution in separ...	**New**

Associated revisions

Revision 20934303 - 03/14/2019 12:17 PM - Alan Gray

Position buffer ops in CUDA

TODO:
- improve the CUDA kernel

Note:
- waits for X copy on the PME stream to finish, need to implement sync point between PME and NB streams (in follow-up).

Implements part of #2817

Change-Id: Ib87dabd74a02727898681249691ac9786b8ac65c

Revision 3cced793 - 03/14/2019 12:19 PM - Alan Gray

F buffer operations in CUDA

TODO: split out reduction

Implements part of #2817

Change-Id: I80c95438e44167b6a9d9d74c27709379f6665867

Revision 42b343b9 - 05/11/2019 01:40 AM - Alan Gray

Position buffer ops in CUDA

On all but search steps the buffer ops transform can now be done on a CUDA GPU. If PME runs on the same GPU the already uploaded coordinates will be used as input.

Activate with GMX_USE_GPU_BUFFER_OPS env variable.

Note:
- waits for X copy on the PME stream to finish, need to implement sync point between PME and NB streams (in follow-up).

Implements part of #2817

Change-Id: Ib87dabd74a02727898681249691ac9786b8ac65c

Revision 3329a50b - 07/11/2019 05:23 PM - Szilárd Páll

Use HostVector for Grid/GridSet data need on-GPU

Grid.cxy_na_, Grid.cxy_ind_, GridSet.cells and GridSet.atomIndices have been converted from std::vector to gmx::HostVector. This allow the code to pin the HostVector when X buffer ops is used and to eliminate the hacky pin/unpin in CUDA buffer ops functions.

Part of #2934
Refs #2817

Change-Id: Icca21dd076128ec582f805ed96e253dfab461270

Revision 8e83edea - 07/19/2019 02:03 PM - Alan Gray

F buffer operations in CUDA

This patch performs GPU buffer ops for force buffers.

Enable with GMX_USE_GPU_BUFFER_OPS env variable.

Currently, the H2D transfer of the force buffer is switched on with haveSpecialForces || haveCpuBondedWork || haveCpuPmeWork, where haveCpuPmeWork is true even when useGpuPme == true until on-GPU PME-nonbonded reduction is added in follow-up.

TODO: enable PME reduction in GPU buffer ops and remove associated H2D transfer

Implements part of #2817

Change-Id: Ice984425301d24bac1340e883698244489cd686e

Revision c8951db1 - 07/22/2019 10:10 AM - Szilárd Páll

Conditionally pin GPU-related grid data

Data that is transferred to the GPU when the buffer ops is offloaded is

now only pinned when the nonbonded module uses GPU offload avoidign the runtime errors encountered when a GPU-enabled build does not detect a GPU and therefore the CUDA runtime refuses to register the memory.

Refs #2817 #2934

Change-Id: Iabbc0d9f37fad0e88cd39a078af1346e8f713ec1

Revision 0a4ca2c4 - 08/15/2019 04:22 PM - Alan Gray

PME reduction for CUDA F buffer operations

Enable with GMX_USE_GPU_BUFFER_OPS env variable.

Provides functionality to perform reduction of PME forces in F buffer ops kernel. Currently active when single GPU performs both PME and PP (multi-GPU support will follow in patch which perfoms PME/PP comms direct between GPUs). When active, Device->Host copy of PME force and CPU-side reduction is disabled.

Implements part of #3029, refs #2817

Change-Id: I3e66b6919c1e86bf0bed42b74136f8694626910b

Revision 120ff490 - 08/16/2019 11:29 AM - Alan Gray

PME reduction for CUDA F buffer operations

Enable with GMX_USE_GPU_BUFFER_OPS env variable.

Provides functionality to perform reduction of PME forces in F buffer ops kernel. Currently active when single GPU performs both PME and PP (multi-GPU support will follow in patch which perfoms PME/PP comms direct between GPUs). When active, Device->Host copy of PME force and CPU-side reduction is disabled.

Implements part of #2817

Change-Id: I3e66b6919c1e86bf0bed42b74136f8694626910b

Revision 741c47d9 - 08/16/2019 11:29 AM - Alan Gray

PME/PP GPU Comms for force buffer

Activate with GMX_GPU_PME_PP_COMMS env variable

Performs scatter of force buffer data from PME task to PP tasks direct to/from GPU memory spaces. Uses direct CUDA memory copies when thread MPI is in use, otherwise CUDA-aware MPI. Uses existing mechanism to perform PME reduction in CUDA F buffer ops function.

Implements part of #2817

Change-Id: I3bf934d20b9af94235532fb030e372af06328a52

Revision 889b6f9a - 09/30/2019 09:18 AM - Szilárd Páll

Make the wait on PME GPU results conditional

When the PME forces are reduced on-GPU and no energy/virial output is produced, we can avoid blocking waiting on the CPU for the PME GPU taks to complete.

This however would break the timing accounting which needs to happen after PME tasks completed. Hence the accounting is moved to the PME output clearing.

Refs #3029, #2817

Change-Id: I4e7f3aa43754a187fe5d6b584803444967516958

Revision 5b594f3b - 10/19/2019 12:52 AM - Alan Gray

GPU Receive for PME/PP GPU Force Communications

This change extends the PME/PP GPU force communication functionality
to allow the force buffer to be recieved direct to GPU memory on the
PP task.

Implements part of #2817
Refs #3158 #3159

Change-Id: I5b1cff1846c7c3bd966b6bf9c0af72769600ef18

Revision c5595a8e - 10/21/2019 11:32 AM - Alan Gray

GPU Coordinate PME/PP Communications

Extends PmePpCommGpu class to provide PP-side support for coordinate
transfers from either GPU or CPU to PME task, and adds new
PmeCoordinateReceiverGpu class to recieve coordinate data directly to
the GPU on the PME task.

Implements part of #2817
Refs TODOs #3157 #3158 #3159

Change-Id: Iefa2bdfd9813282ad8b07feeb7691f16880e61a2

History

#1 - 12/21/2018 05:56 PM - Szilárd Páll

- Description updated

#2 - 02/07/2019 04:06 PM - Szilárd Páll

- Description updated

- Status changed from New to Accepted

#3 - 02/19/2019 02:03 PM - Gerrit Code Review Bot

Gerrit received a related patchset '5' for Issue #2817.
Uploader: Szilárd Páll (pall.szilard@gmail.com)
Change-Id: gromacs~master~Ib87dabd74a02727898681249691ac9786b8ac65c
Gerrit URL: https://gerrit.gromacs.org/9169

#4 - 02/19/2019 02:03 PM - Gerrit Code Review Bot

Gerrit received a related patchset '3' for Issue #2817.
Uploader: Szilárd Páll (pall.szilard@gmail.com)
Change-Id: gromacs~master~I80c95438e44167b6a9d9d74c27709379f6665867
Gerrit URL: https://gerrit.gromacs.org/9170

#5 - 03/06/2019 03:28 PM - Alan Gray

Regarding splitting this up: I am starting to work on a new patch which implements only the GPU Force buffer ops, without the PME reduction.

#6 - 03/08/2019 10:16 AM - Gerrit Code Review Bot

Gerrit received a related patchset '1' for Issue #2817.
Uploader: Alan Gray (alang@nvidia.com)
Change-Id: gromacs~master~Ice984425301d24bac1340e883698244489cd686e
Gerrit URL: https://gerrit.gromacs.org/9275

#7 - 03/08/2019 03:39 PM - Szilárd Páll

Alan Gray wrote:

> Regarding splitting this up: I am starting to work on a new patch which implements only the GPU Force buffer ops, without the PME reduction.

Have you considered also decoupling the GPU-side reduction with CPU forces? This would allow the change to go in independently from all other changes required. Also, if you keep the reduction, do consider changes I76e4b954b4ef045f299a8496b4975497720f4b89 (https://gerrit.gromacs.org/#/c/9126/) and Ie49c0fc483b274ac17e6ace9ca495c11dc719532 (must be a draft).

#8 - 03/08/2019 04:29 PM - Alan Gray

OK, thanks - I will think about this and adjust the new patch accordingly.

#9 - 03/11/2019 11:05 AM - Mark Abraham

I tried to add links to Szilard's text, but one of the patches is a draft

#10 - 03/12/2019 05:02 PM - Szilárd Páll

Mark Abraham wrote:

> I tried to add links to Szilard's text, but one of the patches is a draft

Yes, this was one of the set of changes I uploaded as suggested improvements to the (then single) buffer ops change. The mini-branch of recommendations were kept as drafts and were never intended to undergo code review, so they would just be noise on the already noise gerrit site. If there is interest I can share them wider, but the content should anyway end up in new changes intended fr review.

#11 - 03/13/2019 11:54 AM - Alan Gray

F buffer ops (without PME reduction) patch now updated to include vectorization within kernel and use of haveCpuForces flag, which is currently always set to "true" such that H2D transfer is activated before buffer op, awaiting force workload patch for avaiability of (haveSpecialForces || haveCpuBondedWork).

I considered further splitting this to separate the buffer ops and reduction, but it doesn't really improve isolation because haveCpuForces would still be required to determine if an extra reduction across GPU forces and CPU forces is required.

Now awaiting review.

#12 - 03/14/2019 12:24 PM - Alan Gray

Now rebased such that (haveSpecialForces || haveCpuBondedWork) is available. But we can't use it just yet to make the H2D transfer conditional until the PME reduction is also integrated into the buffer ops. So still setting haveCpuForces=true for now.

#13 - 06/17/2019 02:32 PM - Szilárd Páll

Issues identified in F buffer ops + reduction code; review and resolution done/pending:

- on virial steps PME forced are needed separately
- for DD separate short-range forces are needed separately

For this reason, I suggest to add *in a separate change*, a code-path/kernel flavor that does out of place short-/long-range force reduction. It would be good to have a child F buf ops redmine too track technical issues with the feature.

#14 - 06/25/2019 04:15 PM - Szilárd Páll

Having looked further into the virial step issues here is what we came up with:

- on virial steps for now *turn off* GPU buffer ops instead of shuffling around data to cater for the CPU-side virial reduction
- next, port the virial reduction to the GPU (likely best called from / fused with the reduction kernel); see current code in sim_util.cpp:calc_virial().

#15 - 07/05/2019 07:18 PM - Szilárd Páll

- Status changed from Accepted to In Progress

- Target version set to 2020-beta1

#16 - 08/12/2019 05:03 PM - Szilárd Páll

- Description updated

#17 - 08/14/2019 05:32 PM - Alan Gray

- Description updated

#18 - 09/24/2019 06:52 PM - Mark Abraham

- Target version changed from 2020-beta1 to 2020-beta2

#19 - 10/22/2019 12:20 AM - Szilárd Páll

- Related to Task #3171: schedule CPU H2D force contribution in separate stream added

#20 - 11/01/2019 03:24 PM - Paul Bauer

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

bump