GROMACS - Feature #2891
Task # 3370 (New): Further improvements to GPU Buffer Ops and Comms

PME/PP GPU communications
03/12/2019 02:06 PM - Alan Gray

| Status: | In Progress |
| Priority: | Normal |
| Assignee: | |
| Category: | mdrun |
| Target version: | |
| Difficulty: | uncategorized |

Description
When utilizing multiple GPUs with a dedicated PME GPU, data must be exchanged between the PME task and the PP tasks. The position buffer is gathered to the PME task from the PP task before the PME operation, and the force array is scattered from the PME task to the PP tasks after the operation. Currently, this is routed through the host CPUs, with PCIe transfers and MPI calls operating on data in CPU memory.

Instead, we can transfer data directly between GPU memory spaces using GPU peer-to-peer communication. Modern MPI implementations are CUDA-aware and support this.

TODO extend to support case where PME is on CPU and PP is on GPU.
TODO extend to case where the force reduction is the CPU and a PME rank uses GPU.

Subtasks:

- 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

Related issues:

- Related to GROMACS - Feature #2915: GPU direct communications
  In Progress
- Related to GROMACS - Feature #3087: enable GPU peer to peer access
  Closed
- Related to GROMACS - Bug #3165: task assignment silent abort
  New
- Related to GROMACS - Task #3195: assess nightly master failures
  Closed
- Related to GROMACS - Task #3360: investigate the future of DD dynamic load ba...
  New

Associated revisions

Revision 4dd80128 - 08/16/2019 11:29 AM - Alan Gray
PME/PP GPU Comms for position buffer
Activate with GMX_GPU_PME_PP_COMMS env variable
Performs gather of position buffer data from PP tasks to PME task with transfers operating directly to/from GPU memory. Uses direct CUDA memory copies when thread MPI is in use, otherwise CUDA-aware MPI.
Implements part of Feature #2891
Change-ID: if6222eccfe30099beeb25a64ccebe318d0a3b1dbc

Revision ec0aa356 - 10/16/2019 12:04 AM - Alan Gray
PME/PP GPU Pimpl Class and GPU->CPU Communication for force buffer
Activate with GMX_GPU_PME_PP_COMMS env variable
Implements new pimpl class for PME-PP GPU communications. Performs scatter of force buffer data from PME task GPU buffer to PP task CPU buffers directly using CUDA memory copies. Requires thread MPI to be in use.
Implements part of #2891

Change-Id: i0181ff67065c75f20cddc361f695df9bf888cd88

History

#1 - 03/12/2019 02:46 PM - Alan Gray
- Description updated

#2 - 03/13/2019 11:56 AM - Alan Gray
Awaiting merge of buffer ops patch.

#3 - 04/02/2019 02:22 PM - Gerrit Code Review Bot
Gerrit received a related patchset ‘1’ for Issue #2891.
Uploader: Alan Gray (alang@nvidia.com)
Change-Id: gromacs~master~If6222eccfe30099beeb25a64cceb318d0a3b1dcbc
Gerrit URL: https://gerrit.gromacs.org/9385

#4 - 04/02/2019 04:45 PM - Szilárd Páll
- Category set to mdrun

Just had a look at the proposed change and I think we should perhaps take the time to discuss some implementation choices here. There apply to all direct GPU communication you are working on, so it may make sense to open a new issue where such general things are discussed?

A few questions to kick off with:

- How do we provide fallbacks for when i) no MPI is used ii) no CUDA-aware MPI is used?
  - For the former, with tMPI I assume we can have a GPUDirect-based fallback.
  - For the latter, how do we detect that we have a CUDA-aware MPI? What happens if we don’t and the proposed code is invoked?
- As noted in CR, we should initiate the PP->PME send exactly at the same location where the CPU path does it; the coordinates are available there so there seems to be no reason to not unify the paths.

#5 - 04/02/2019 05:30 PM - Alan Gray

As noted in CR, we should initiate the PP->PME send exactly at the same location where the CPU path does it; the coordinates are available there so there seems to be no reason to not unify the paths.

Yes, agreed.

Moving other Q to new issue 2915

#6 - 04/03/2019 02:59 PM - Szilárd Páll
- Related to Feature #2915: GPU direct communications added

#7 - 09/12/2019 04:24 PM - Szilárd Páll
- Related to Feature #3087: enable GPU peer to peer access added

#8 - 09/19/2019 05:21 PM - Alan Gray
- Description updated

#9 - 09/20/2019 01:08 AM - Szilárd Páll
> TODO extend to support case where PME is on CPU and PP is on GPU.

Similar functionality would be needed if we’d want to support PME mixed mode on a separate PME rank.

#10 - 10/09/2019 03:51 PM - Szilárd Páll
- Description updated

#11 - 10/15/2019 11:18 PM - Szilárd Páll
The code implementing this feature was broken by:
https://gerrit.gromacs.org/c/gromacs/+/13437
This needs fixing before we can use unit tests to run this feature at all.
The code implementing this feature was broken by:
https://gerrit.gromacs.org/c/gromacs/+/13437
This needs fixing before we can use unit tests to run this feature at all.

It was revealed by the aforementioned change to the tests, not broken by it.

- Description updated

- Related to Bug #3165: task assignment silent abort added

- Related to Task #3195: assess nightly master failures added

- Related to Task #3360: investigate the future of DD dynamic load balancing with GPU offload added

- Status changed from New to In Progress

- Parent task changed from #2816 to #3370