Implementing Open SYCL support for oneMKL

Bálint Soproni

Heidelberg University
oneAPI Academic Center of Excellence
Open SYCL project

March 8, 2023

Introduction

- Msc Student at University Heidelberg
- Student Assistant 2019-2022
- Implementing Open SYCL support for oneMKL in 2021
- Since 2022 at StreamHPC, some GROMACS contributions

Introduction Open SYCL

- Open source SYCL implementation developed at Heidelberg University
- Multiple compilation flows, targeting NVIDIA, AMD, Intel GPUs and CPUs
- Leverages existing toolchains
- Until recently called hipSYCL

Task

- Add support for the existing RNG and BLAS backends via Open SYCL
  - cuBLAS
  - MKLCPU
  - cuRAND
- Enable targeting AMD GPUs for RNG and BLAS domains with Open SYCL
  - rocRAND
  - rocBLAS

Implementation process

- Compile the repository with Open SYCL
  - Abstract the implementation-specific parts
- Fix issues surfacing with hipSYCL
- Implement wrappers for AMD GPU libraries

Improve cmake integration with other SYCL implementations

- Open SYCL uses (similiarly to computeCPP and triSYCL) the `add_sycl_to_target` function for cmake integration
  - Add conditional support
- Add option to specify `ONEMKL_SYCL_IMPLEMENTATION`

Improvements for generality

Optional half support

- Half is optional according to the SYCL specification
- Open SYCL did not implement half
- Use SYCL idioms to query support

Improvements for generality

Optional half support

- Half is optional according to the SYCL specification
- Open SYCL did not implement half
- Use SYCL idioms to query support

Backend interoperability

- Interop_task is not standard SYCL
- Use `onemkl_cublas_host_task`
- Enables switching between hipSYCL and DPC++ host interop functionality

Improvements for generality

Use appropriately qualified names

- Ambiguous kernel names
- Issue with the declared namespaces
- Open SYCL only declares `sycl::` in sycl/sycl.hpp, and `::cl::sycl` in CL/sycl.hpp
- Specification only guarantees one of the namespaces to be present

```
#if __has_include(<sycl/sycl.hpp>)
#include <sycl/sycl.hpp>
#else
#include <CL/sycl.hpp>
#endif
```

Uncovered runtime issues

oneMKL

- After a level1 call, all subsequent calls failed
  - CUBLAS_POINTER_MODE_DEVICE was set
  - but it was never reset to CUBLAS_POINTER_MODE_HOST
- Some missing synchronizations were fixed for iamin, iamax

ROCrand

- There were issues with skipping for some generators

Conclusion

- OneMKL could be a key quality of life feature for SYCL developers
- Allows applications to save writing boilerplate wrappers
  - in GROMACS 8 files and 1000+ lines of code
- Testing with multiple compilers is beneficial
- Implementing CI is crucial to ensure functionality

Conclusion

- OneMKL could be a key quality of life feature for SYCL developers
- Allows applications to save writing boilerplate wrappers
  - in GROMACS 8 files and 1000+ lines of code
- Testing with multiple compilers is beneficial
- Implementing CI is crucial to ensure functionality

Issues for adoption

- oneMKL does not ship with the oneAPI toolkit
- Additional dependency, which needs to be installed by the user
- Installation is perceived to be difficult
- Library naming is confusing
- Need for more backends: rocFFT, cuFFT are not always the most performant

Questions/Limitations

- Multiple backends for the same device would be beneficial
- Would it be possible to add rocFFT, VkFFT, dbFFT?
- How to handle two libraries for the same backend?
- Could `add_sycl_to_target` be implemented for DPC++?

Questions/Limitations

- Multiple backends for the same device would be beneficial
- Would it be possible to add rocFFT, VkFFT, dbFFT?
- How to handle two libraries for the same backend?
- Could `add_sycl_to_target` be implemented for DPC++?

Special tahnks to

- Aksel Alpay
- Maria Kraynyuk & Mesut Meterelliyoz
- Sanchi Vaishnavi & Nils Friess