oneMKL Technical Advisory Board

Session 3

June 17, 2020

Agenda

- Welcoming remarks – 5 minutes
- Updates from last meeting – 5 minutes
- Finish walk-thru of the oneMKL Specification – Spencer Patty (10 minutes)
- Overview of the open source oneMKL interfaces GitHub project – Julia Sukharina (20 minutes)
- Wrap-up and next steps – 5 minutes

Updates from last meeting

- Mark Hoemmen started a discussion on dense linear algebra functions needing encapsulations for matrices and vectors
  - Following up with DPC++ team on non-contiguous sub-buffer support
- John Pennycook looked at mdspan with SYCL as a potential route to providing accessor-like functionality on top of USM pointers
- New official release of the specification announced: version 0.8.5

Version	Date	Notes
0.8.0	05/28/2020	80% content
0.8.5	06/26/2020	85% content
0.9.0	07/30/2020	Final Gold Preview
1.0.0	08/30/2020	Gold Release

(Finish) Walk-thru of the oneMKL Specification

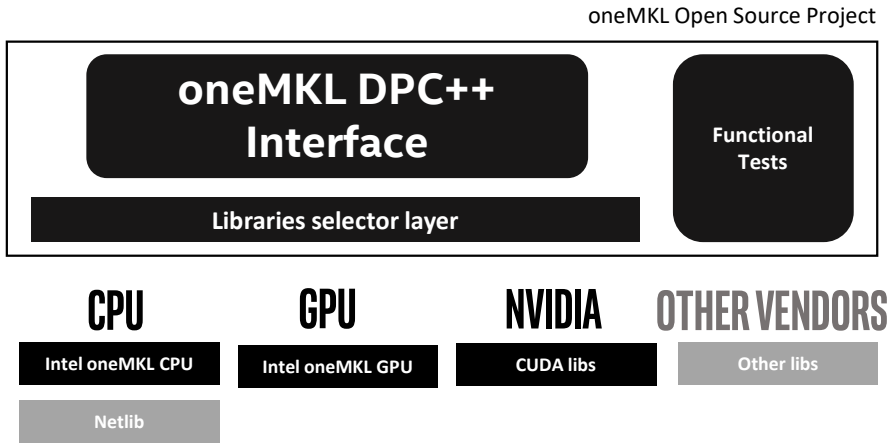
https://spec.oneapi.com/versions/latest/elements/oneMKL/source/index.html

Overview of the open source oneMKL interfaces GitHub project

https://github.com/oneapi-src/oneMKL

Overview

- oneMKL DPC++ Interfaces open source project implements oneMKL Specification. Interfaces can change more rapidly than interfaces in Intel oneMKL binary product.
- Current status:
  - BLAS interfaces: buffer and Unified Shared Memory (USM) APIs
  - CPU, GPU and Nvidia support
  - Linux and Windows
  - Possibility to download dependencies via Conan package manager (Linux only)



- Performance is provided by linking to vendor optimized libraries underneath oneMKL DPC++ API (Intel oneMKL, cuBLAS).
- Binary libraries are not part of open source and should be downloaded by user separately.

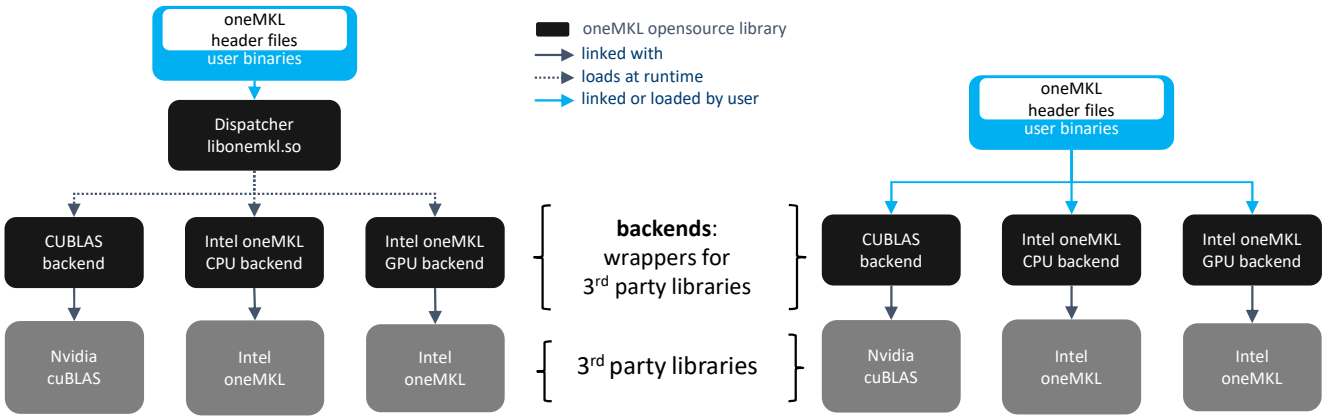
Two Types of Interfaces

Auto backend selection

The application is linked with the oneMKL dispatcher library and the required backend is loaded at run-time based on queue.

Manual backend selection

The application uses a templated API to specify the required device and third-party library, and the application is linked directly with the required oneMKL backend.



- Examples of current API

```
// use MKLCPU backend
gemm<intelcpu, intelmkl>(q,…);
// use MKLGPU backend
gemm<intelgpu, intelmkl>(q,…);
// use CUBLAS backend
gemm<nvidiagpu, cublas>(q,…);
// use NETLIB backend
gemm<intelcpu, netlib>(q,…);
```

- Changes to be implemented in a month (names are subject to change)

```
// use MKLCPU backend for Intel oneMKL
gemm<mklcpu>(q,…);
// use MKLGPU backend for Intel oneMKL
gemm<mklgpu>(q,…);
// use CUBLAS backend for NV cuBLAS
gemm<cublas>(q,…);
// use NETLIB backend for Netlib x86
gemm<netlib>(q,…);
```

Next Steps

- Look over current oneMKL Spec v. 0.8
  - Focus most on oneMKL Architecture (section 1) and BLAS/LAPACK APIs
- Focuses for next meeting(s):
  - Particular feedback requests on oneMKL Spec:
    - Exceptions/error handling
    - Asynchronous execution

Resources

- oneAPI Main Page: https://www.oneapi.com/
- Latest release of oneMKL Spec (currently v. 0.8): https://spec.oneapi.com/versions/latest/elements/oneMKL/source/index.html
- GitHub for oneAPI Spec: https://github.com/oneapi-src/oneAPI-spec
- GitHub for oneAPI TAB: https://github.com/oneapi-src/oneAPI-tab
- Latest build of oneAPI Spec: http://staging.spec.oneapi.com.s3-website-us-west-2.amazonaws.com/exclude/ci/branches/refs/heads/master/versions/latest/index.html

- GitHub for open source oneMKL interfaces (currently BLAS domain): https://github.com/oneapi-src/oneMKL