# oneMKL Technical Advisory Board

Session 18

June 8, 2022

# Agenda

- Welcoming remarks – 5 minutes
- Updates from last meeting – 5 minutes
- Update on the oneMKL interfaces open source project – Mesut Meterelliyoz (30 minutes)
- Wrap-up and next steps – 5 minutes

# Updates from last meeting

- Question from last meeting: process for deprecating older versions of the specification:
  - We don’t deprecate the spec, but we can deprecate an API in a spec. oneAPI v1.1 is a superset of v1.0.
- Blog post by Aksel Alpay: Heidelberg University drives heterogenous computing with oneAPI
  - oneMKL Interfaces RNG and BLAS domain now support AMD GPUs
- Intel oneMKL 2022.1 was released

# Update on the oneMKL open-source interfaces project

# oneMKL interfaces - Overview

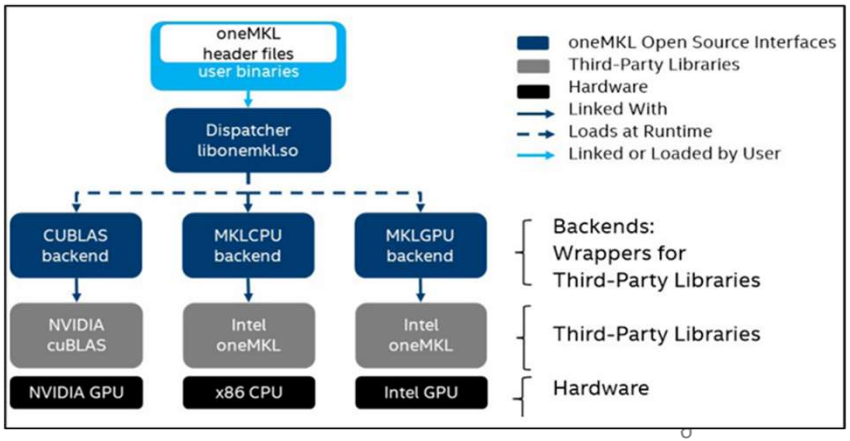
- Presented by Maria Kraynyuk 2 years ago
- oneMKL interfaces project enables to achieve best possible performance on different hardware architectures using the **same** source code.
  - No need to re-write source code for different accelerators
  - Supports both CPUs and GPUs
  - Supports both Intel and Nvidia GPUs
- Started with BLAS domain.

# oneMKL interfaces – Usage Models

- Run-time dispatch
- Dispatcher: Select and load third-party library based on the device information
- Backend wrapper: Convert DPC++ run-time objects from the application to third party-library specific objects and call third-party function.
- Supports only dynamic link

```
//app.cpp
#include "oneapi/mkl.hpp“
…
cpu_dev = sycl::device(sycl::cpu_selector());
gpu_dev = sycl::device(sycl::gpu_selector());
sycl::queue cpu_queue(cpu_dev);
sycl::queue gpu_queue(gpu_dev);
oneapi::mkl::blas::column_major::gemm(cpu_queue, transA, transB, m, ...);
oneapi::mkl::blas::column_major::gemm(gpu_queue, transA, transB, m, …);
```

```
$> dpcpp -fsycl –I$ONEMKL/include app.cpp
$> dpcpp -fsycl app.o –L$ONEMKL/lib –lonemkl
```



# oneMKL interfaces – Usage Models

- Compile-time dispatch
- The application uses a templated backend selector API
- The application is linked with the required backend wrapper library.
- Supports both static and dynamic link

```
//app.cpp
#include "oneapi/mkl.hpp“
…
cpu_dev = sycl::device(sycl::cpu_selector());
gpu_dev = sycl::device(sycl::gpu_selector());
sycl::queue cpu_queue(cpu_dev);
sycl::queue gpu_queue(gpu_dev);
oneapi::mkl::backend_selector<oneapi::mkl::backend::mklcpu> cpu_selector(cpu_queue);
oneapi::mkl::backend_selector<oneapi::mkl::backend::cublas> gpu_selector(gpu_queue);
oneapi::mkl::blas::column_major::gemm(cpu_selector, transA, transB, m, ...);
oneapi::mkl::blas::column_major::gemm(gpu_selector, transA, transB, m, ...);
```

```
$> clang++ -fsycl –I$ONEMKL/include app.cpp
$> clang++ -fsycl app.o –L$ONEMKL/lib –lonemkl_blas_mklcpu –lonemkl_blas_cublas
```

# oneMKL interfaces – What is new?

- New domains: RNG and LAPACK
- New hardware: AMD GPUs
- New backends: NETLIB, cuSOLVER, cuRAND, rocBLAS, rocRAND
- New compiler: hipSYCL
- Align with SYCL2020 specification
- Added examples for all domains
- Close performance gap between cuBLAS backend and native cuBLAS library
- Added new BLAS APIs and functional tests to align with oneMKL spec
- Bug fixes and documentation updates

# Support Matrix by compiler

| DPC++ | | | |
|---|---|---|---|
| OS | Domain | Hardware | Third-party library |
| Linux | BLAS | x86 CPU | Intel oneMKL |
| Linux | BLAS | Intel GPU | Intel oneMKL |
| Linux | BLAS | x86 CPU | LAPACK NETLIB |
| Linux | LAPACK | x86 CPU | Intel oneMKL |
| Linux | LAPACK | Intel GPU | Intel oneMKL |
| Linux | RNG | x86 CPU | Intel oneMKL |
| Linux | RNG | Intel GPU | Intel oneMKL |
| Windows | BLAS | x86 CPU | Intel oneMKL |
| Windows | BLAS | Intel GPU | Intel oneMKL |
| Windows | BLAS | x86 CPU | LAPACK NETLIB |
| Windows | LAPACK | x86 CPU | Intel oneMKL |
| Windows | LAPACK | Intel GPU | Intel oneMKL |
| Windows | RNG | x86 CPU | Intel oneMKL |

| LLVM | | | |
|---|---|---|---|
| OS | Domain | Hardware | Third-party library |
| Linux | BLAS | x86 CPU | Intel oneMKL |
| Linux | BLAS | x86 CPU | LAPACK NETLIB |
| Linux | BLAS | NVIDIA GPU | NVIDIA cuBLAS |
| Linux | BLAS | AMD GPU | AMD rocBLAS * |
| Linux | LAPACK | x86 CPU | Intel oneMKL |
| Linux | LAPACK | NVIDIA GPU | NVIDIA cuSOLVER |
| Linux | RNG | x86 CPU | Intel oneMKL |
| Linux | RNG | NVIDIA GPU | NVIDIA cuRAND |
| Windows | BLAS | x86 CPU | Intel oneMKL |
| Windows | BLAS | x86 CPU | LAPACK NETLIB |
| Windows | LAPACK | x86 CPU | Intel oneMKL |
| Windows | RNG | x86 CPU | Intel oneMKL |

→ Codeplay

→ Codeplay

→ Berkeley National Laboratory (article)

* PR in progress

| hipSYCL | | | |
|---|---|---|---|
| OS | Domain | Hardware | Third-party library |
| Linux | BLAS | x86 CPU | Intel oneMKL |
| Linux | BLAS | x86 CPU | LAPACK NETLIB |
| Linux | BLAS | AMD GPU | AMD rocBLAS |
| Linux | BLAS | NVIDIA GPU | NVIDIA cuBLAS * |
| Linux | RNG | x86 CPU | Intel oneMKL |
| Linux | RNG | AMD GPU | AMD rocRAND |
| Linux | RNG | NVIDIA GPU | NVIDIA cuRAND * |

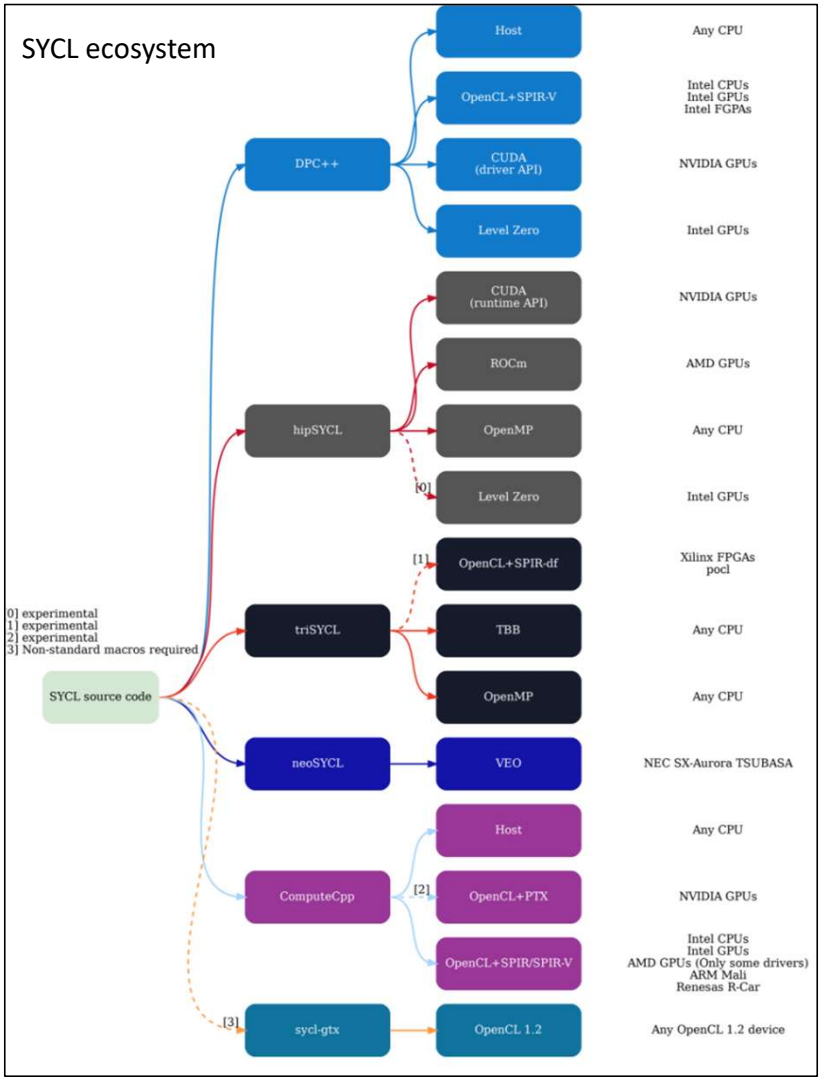
Heidelberg University (article)

* PR in progress

DPC++: https://www.intel.com/content/www/us/en/developer/tools/oneapi/base-toolkit-download.html
LLVM: https://github.com/intel/llvm
hipSYCL: https://github.com/illuhad/hipSYCL

# hipSYCL - Overview

- hipSYCL is an open source SYCL implementation led by Aksel Alpay at Heidelberg University
- Does not rely on OpenCL, uses existing toolchains directly to target different hardware.
- Not all hardware vendors have adopted SYCL yet, so hipSYCL aims to close that gap.
- Enables SYCL code to utilize vendor optimized libraries as well as vendor debug and performance tuning tools.



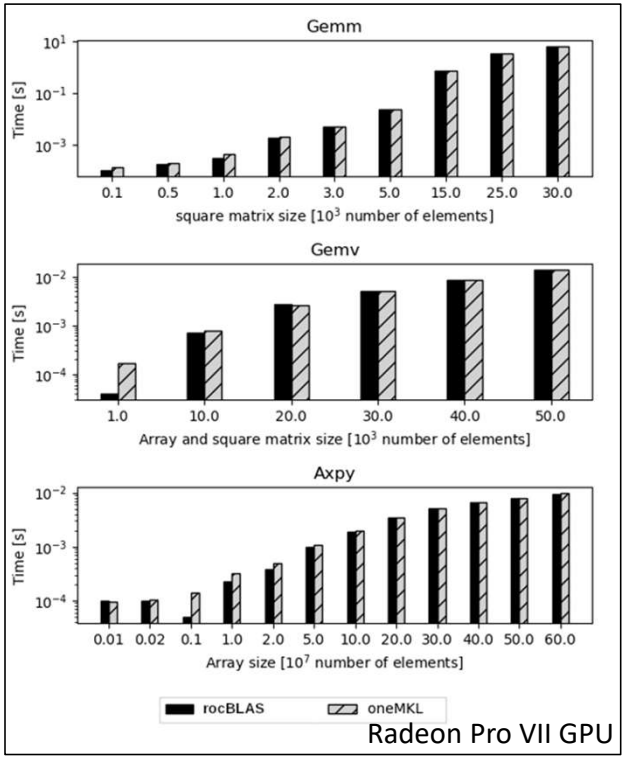
# hipSYCL – Center of Excellence program

- Intel has 22 oneAPI Center of Excellence (CoE) programs with several national laboratories and universities around globe.
- CoE with Heidelberg Univ:
  - Bring oneAPI components to AMD hardware by leveraging hipSYCL
  - First and only attempt to implement oneAPI with a compiler independent of DPC++
  - **Programming Model:** Support key SYCL 2020 features to compile oneAPI code and achieve within 80% of CUDA/ROCm performance
  - **Libraries:** Run oneAPI libraries – use oneMKL interfaces as a proof point
  - **Building blocks:** Low-level API Level-Zero must be supported.
- See IWOCL 2022 article for more details:
  - Exploring the possibility of a hipSYCL-based implementation of oneAPI

# hipSYCL – oneMKL interfaces

- oneMKL interfaces can be built with hipSYCL and target x86 CPUs as well as AMD GPUs.
- Both BLAS and RNG domains are supported.
- Performance comparison: oneMKL interfaces achieves similar performance as native rocBLAS calls
- Future Work:
  - Add NVIDIA GPU support with hipSYCL (PR open)
  - Add LAPACK domain support with hipSYCL
  - Add Intel GPU support with hipSYCL through Level-Zero

| hipSYCL | | | |
|---|---|---|---|
| OS | Domain | Hardware | Third-party library |
| Linux | BLAS | x86 CPU | Intel oneMKL |
| Linux | BLAS | x86 CPU | LAPACK NETLIB |
| Linux | BLAS | AMD GPU | AMD rocBLAS |
| Linux | BLAS | NVIDIA GPU | NVIDIA cuBLAS * |
| Linux | RNG | x86 CPU | Intel oneMKL |
| Linux | RNG | AMD GPU | AMD rocRAND |
| Linux | RNG | NVIDIA GPU | NVIDIA cuRAND * |



Radeon Pro VII GPU

# Wrap-up

# Next Steps

- Focuses for next meeting(s):
  - Device APIs for BLAS
  - Any topics from oneMKL TAB members?

- If anyone has content that they would like posted on oneAPI.io, please let us know

# Resources

- oneAPI Main Page: https://www.oneapi.io/
- Latest release of oneMKL Spec (currently v. 1.1): 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

- GitHub for open source oneMKL interfaces (currently BLAS, RNG, and LAPACK domains): https://github.com/oneapi-src/oneMKL