oneMKL Technical Advisory Board

Session 6

August 12, 2020

Agenda

- Welcoming remarks – 5 minutes
- Updates from last meeting – 10 minutes
- oneMKL Random Number Generators pass downs and open questions - Pavel Dyakov and Alina Elizarova (15 minutes)
- Overview of oneMKL Summary Statistics domain – Pavel Dyakov and Alina Elizarova (15 minutes)
- Wrap-up and next steps – 5 minutes

oneMKL Specification v. 0.9

- Released July 30!
- Modifications to oneMKL Architecture, BLAS and LAPACK domains
- Significant refactoring and updating of Sparse BLAS, VM, RNG, and DFT domains API descriptions and structure
- Add Summary Statistics domain
- Add future considerations and acknowledgment to appendices
- Change top-level namespace to oneapi::mkl

oneMKL Specification v. 1.0 and later

- Targeting v. 1.0 (release date August 30):
    - Exceptions
    - Minor RNG changes
    - Formatting, typo fixes
- Considered for future versions:
    - Encapsulation of matrix and vector objects
    - More human-readable names
    - Broader support for row major layout
    - Alternative handling of computational failures

oneMKL TAB Meeting Frequency

- Will present overviews of more domains:
  - Sparse linear algebra
  - Discrete Fourier transforms
  - Vector math
- Any requests for other topics?
- Change frequency to every 4 weeks?

oneMKL Random Number Generators Pass Downs and Open Questions

Actions Taken from Previous Meeting for oneMKL RNG

- All trivial constructors for distributions are removed from specification (0.9)
- Keep engines with copy constructor (deep-copy) and add move constructor (will be updated in 1.0)

oneMKL RNG Engines

- `oneapi::mkl::rng::mrg32k3a`
- `oneapi::mkl::rng::philox4x32x10`
- `oneapi::mkl::rng::mt19937`
- `oneapi::mkl::rng::mt2203`

Reliable, performable, well-known and widely used

- `oneapi::mkl::rng::mcg31m1`
- `oneapi::mkl::rng::mcg59`

Small period, but quite fast and usable

- `oneapi::mkl::rng::ars5`
- `oneapi::mkl::rng::nondeterministic`
- `oneapi::mkl::rng::sfmt19937`

Performance and/or implementation is HW dependent

- `oneapi::mkl::rng::whichmann_hill`
- `oneapi::mkl::rng::r250`

Popular before, but outdated

- `oneapi::mkl::rng::sobol`
- `oneapi::mkl::rng::niederreiter`

Quasi-random

Open questions:

- Is “default_engine” needed?

  Proposal: *Adding "default_engine" to oneMKL spec as "implementation defined" (to support users who don’t care about underlying engines)*

- HW dependent engines: should we have such engines in oneMKL spec?

  Proposal: *Keeping HW dependent engines in oneMKL spec (to provide performable engines for the particular devices)*

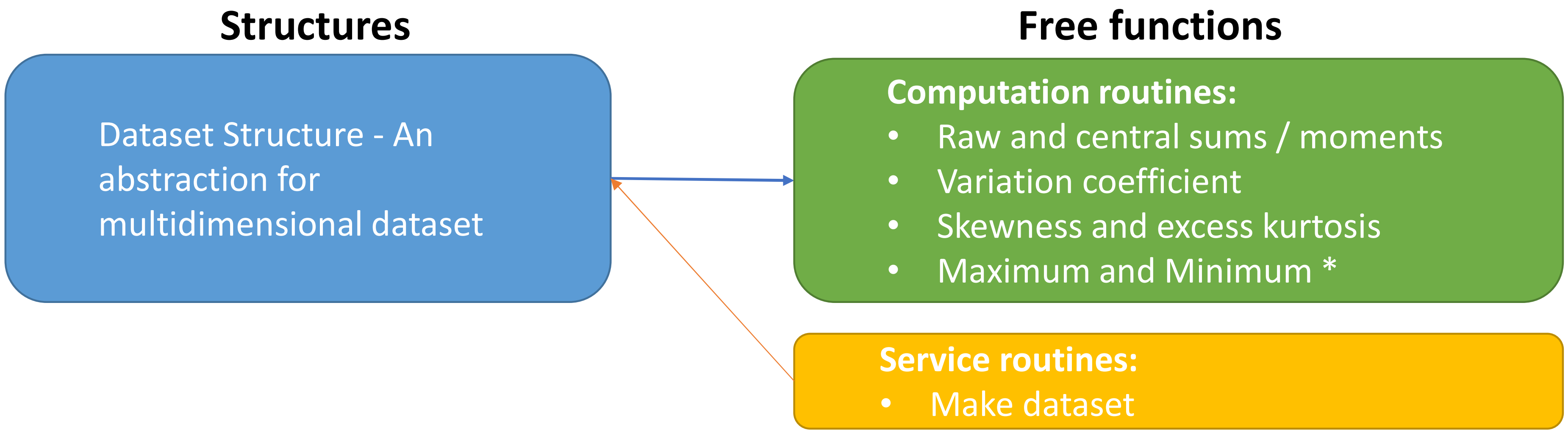
- Outdated engines: do we want to keep these engines in oneMKL spec?

  Proposal: *Keeping outdated engines in oneMKL spec (to provide wide set of engines)*

Overview of oneMKL Summary Statistics Domain

oneMKL Summary Statistics Overview and Structure

- oneMKL Summary Statistics provide a set of routines that compute basic statistical estimates for single and double precision datasets.



Structures

Dataset Structure - An abstraction for multidimensional dataset

Free functions

Computation routines:

- Raw and central sums / moments
- Variation coefficient
- Skewness and excess kurtosis
- Maximum and Minimum *

Service routines:

- Make dataset

* Routines set is targeted to be extended after 1.0 specification version

oneMKL Summary Statistics Usage Model

A typical algorithm for summary statistics is as follows:

- Create and initialize an object for dataset.
- Call the summary statistics routine to calculate the appropriate estimate.

* currently 1D buffers and USM pointers are used to represent multi-dimensional dataset

```
#include "CL/sycl.hpp"
#include "mkl_stats_sycl.hpp"

int main() {
    sycl::queue queue;

    const size_t n_observations = 1000;
    const size_t n_dims = 3;
    std::vector<float> x(n_observations * n_dims);
    // [fill x storage]

    //create buffer for dataset
    sycl::buffer<float, 1> x_buf(x.data(), x.size());
    // create buffer for mean values
    sycl::buffer<float, 1> mean_buf(n_dims);
    // create oneapi::mkl::stats::dataset
    auto dataset = oneapi::mkl::stats::make_dataset<oneapi::mkl::
stats::layout::row_major>(n_dims, n_observations, x_buf);

    oneapi::mkl::stats::mean(queue, dataset, mean_buf);

    //...
    return 0;
}
```

oneMKL Summary Statistics Dataset Structure

- Supported layout:
  - `oneapi::mkl::stats::row_major`
  - `oneapi::mkl::stats::col_major`

Parameter	Description
`n_dims`	the number of dimensions
`n_observations`	the number of observations
`observations`	the matrix of observations
`weights`	an optional parameter, represents array of weights for observations (of size n_observations). If the parameter is not specified, each observation is assigned a weight equal 1.
`indices`	an optional parameter, represents array of dimensions that are processed (of size n_dims). If the parameter is not specified, all dimensions are processed.

```
// Buffer-based
template<typename Type, layout ObservationsLayout>
struct dataset<sycl::buffer<Type, 1>, ObservationsLayout> {
    explicit dataset(std::int64_t n_dims_,
           std::int64_t n_observations_,
           sycl::buffer<Type, 1> observations_,
           sycl::buffer<Type, 1> weights_ = {0},
           sycl::buffer<std::int64_t, 1> indices_ = {0});

    std::int64_t n_dims;
    std::int64_t n_observations;
    layout = ObservationsLayout;
    sycl::buffer<Type, 1> observations;
    sycl::buffer<Type, 1> weights;
    sycl::buffer<std::int64_t, 1> indices;
};

// USM based
template<typename Type, layout ObservationsLayout>
struct dataset<Type*, ObservationsLayout> {
    explicit dataset(std::int64_t n_dims_,
           std::int64_t n_observations_,
           Type* observations_, Type* weights_ = nullptr,
           std::int64_t* indices_ = nullptr);

    std::int64_t n_dims;
    std::int64_t n_observations;
    layout = ObservationsLayout;
    Type* observations;
    Type* weights;
    std::int64_t* indices;
};
```

oneMKL Summary Statistics Service Routines Make Dataset

Service function to create a dataset object (for C++ version < 17 to avoid explicit USM/buffer template parameter instantiation)

Supported types:

- `float`
- `double`

```
// Buffer-based API. Make dataset
template<layout ObservationsLayout = row_major, typename Type>
dataset<sycl::buffer<Type, 1>, ObservationsLayout>
make_dataset(std::int64_t n_dims, std::int64_t n_observations,
          sycl::buffer<Type, 1> observations,
          sycl::buffer<Type, 1> weights = {0},
          sycl::buffer<std::int64_t, 1> indices = {0});

// USM-based API. Make dataset
template<layout ObservationsLayout = layout::row_major, typename Type>
dataset<Type*, ObservationsLayout>
make_dataset(std::int64_t n_dims, std::int64_t n_observations,
          Type* observations,
          Type* weights = nullptr,
          std::int64_t* indices = nullptr);
```

oneMKL Summary Statistics Routines Mean / Skewness / Variation / Kurtosis / Min / Max

Compute mean / skewness* / variation* / kurtosis* / min / max values

Supported types:

- `float`
- `double`

Supported methods:

- `oneapi::mkl::stats::method::fast`
- `oneapi::mkl::stats::method::one_pass**`

* Support user provided mean case

** Not supported for min / max and skewness / variation / kurtosis/ with user provided mean

```
// Buffer-based API. Mean
template<method Method = method::fast, typename Type,
layout ObservationsLayout>
void mean(sycl::queue& queue,
          const dataset<sycl::buffer<Type, 1>, ObservationsLayout>&
data,
          sycl::buffer<Type, 1> mean);

// USM-based API. Mean
template<method Method = method::fast, typename Type,
layout ObservationsLayout>
sycl::event mean(sycl::queue& queue,
          const dataset<Type*, ObservationsLayout>& data,
          Type* mean,
          const sycl::vector_class<sycl::event> &dependencies = {});
```

oneMKL Summary Statistics Routines Central Sums and Moments

Compute central sums (moments) up to the 4th order

Supported types:

- `float`
- `double`

Supported methods:

- `oneapi::mkl::stats::method::fast`
- `oneapi::mkl::stats::method::one_pass`

```
// Buffer-based API. Central sums
template<method Method = method::fast, typename Type,
layout ObservationsLayout>
void raw_sum(sycl::queue& queue,
           const dataset<sycl::buffer<Type, 1>, ObservationsLayout>&
data,
           sycl::buffer<Type, 1> central_sum_2,
           sycl::buffer<Type, 1> central_sum_3 = {0},
           sycl::buffer<Type, 1> central_sum_4 = {0});

// USM-based API. Central sums
template<method Method = method::fast, typename Type,
layout ObservationsLayout>
sycl::event raw_sum(sycl::queue& queue,
           const dataset<Type*, ObservationsLayout>& data,
           Type* central_sum_2,
           Type* central_sum_3 = nullptr,
           Type* central_sum_4 = nullptr,
           const sycl::vector_class<sycl::event> &dependencies = {});
```

Next Steps

- Look over current oneMKL Spec v. 0.9
- Focuses for next meeting(s):
    - Sparse linear algebra
    - Discrete Fourier transforms
    - Vector math

Version	Date	oneAPI Notes	oneMKL Notes
0.8.5	26 June 2020	80% content	Finalize BLAS and LAPACK domains
0.9.0	30 Jul 2020	~100% content	Finalize FFT, sparse BLAS, RNG, and VM domains
1.0.0	30 Aug 2020	Gold Release	Minor cleanup

Resources

- oneAPI Main Page: https://www.oneapi.com/
- Latest release of oneMKL Spec (currently v. 0.9): 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