# oneMKL Technical Advisory Board

Session 2

June 3, 2020

## Agenda

- Welcoming remarks – 5 minutes
- Overview of oneMKL programming model – Maria Kraynyuk (25 minutes)
- Walk-thru of the oneMKL Specification – Spencer Patty (20 minutes)
- Wrap-up and next steps – 5 minutes

## oneMKL TAB Members

- Mike Dewar, Numerical Algorithms Group (NAG)
- Mark Hoemmen, Stellar Science
- Nevin Liber, Argonne National Laboratory (ANL)
- Piotr Luszczek, Innovative Computing Laboratory (ICL) at University of Tennessee, Knoxville (UTK)
- Pat Quillen, MathWorks
- Nichols Romero, ANL
- Harry Waugh, University of Bristol

## oneAPI Math Kernel Library Programming Model

## Example: Matrix Multiplication (1 of 3)

**Buffer API**

```
int64_t n = 32;

sycl::device dev(sycl::{host,cpu,gpu}_selector());
sycl::queue Q(dev);

double *A, *B, *C;
A = (double *)malloc(n*n*sizeof(double));
...

sycl::buffer<double, 1> A_buf{A, range<1>(n * n)},
                        B_buf{B, range<1>(n * n)},
                        C_buf{C, range<1>(n * n)};

onemkl::blas::gemm(Q,
      onemkl::transpose::N, onemkl::transpose::N,
      n, n, n, 1.0,
      A_buf, n,
      B_buf, n, 0.0,
      C_buf, n);
```

device setup

prepare matrices

C = A * B

**C API**

```
int64_t n = 32;



double *A, *B, *C;
A = (double *)malloc(n*n*sizeof(double));
...

cblas_dgemm(CblasRowMajor,
      CblasNoTrans, CblasNoTrans,
      n, n, n, 1.0,
      A, n,
      B, n, 0.0,
      C, n);
```

# Example: Matrix Multiplication (2 of 3)

**Buffer API**

```
using onemkl::blas::gemm;
int64_t n = 32;
sycl::device dev(sycl::{host,cpu,gpu}_selector());
sycl::queue Q(dev);


double *A = ..., *B = ..., *C = ...;

sycl::buffer<double, 1> A_buf{A, range<1>(n * n)},
                        B_buf{B, range<1>(n * n)},
                        C_buf{C, range<1>(n * n)};



gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
        n, n, n, 1.0, A_buf, n, B_buf, n,
        0.0, C_buf, n);
```

device setup

prepare matrices

C = A * B

**USM API**

```
using onemkl::blas::gemm;
int64_t n = 32;
sycl::device dev(sycl::{host,cpu,gpu}_selector());
sycl::queue Q(dev);


size_t bytes = n * n * sizeof(double);
double *A = sycl::malloc_shared(bytes, dev,
                          Q.get_context());
double *B = sycl::malloc_shared(…);
double *C = sycl::malloc_shared(…);



gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
     n, n, n, 1.0, A, n, B, n,
     0.0, C, n);

Q.wait_and_throw();
```

# Example: Matrix Multiplication (3 of 3)

**Buffer API**

```
using onemkl::blas::gemm;
int64_t n = 32;
sycl::device dev(sycl::{host,cpu,gpu}_selector());
sycl::queue Q(dev);


double *A = ..., *B = ..., *C = ...;

sycl::buffer<double, 1> A_buf{A, range<1>(n * n)},
                        B_buf{B, range<1>(n * n)},
                        C_buf{C, range<1>(n * n)},
                        D_buf{D, range<1>(n * n)};


gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
        n, n, n, 1.0, A_buf, n, B_buf, n,
        0.0, C_buf, n);

gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
        n, n, n, 1.0, C_buf, n, A_buf, n,
        0.0, D_buf, n);
```

device setup

prepare matrices

C = A * B

D = C * A

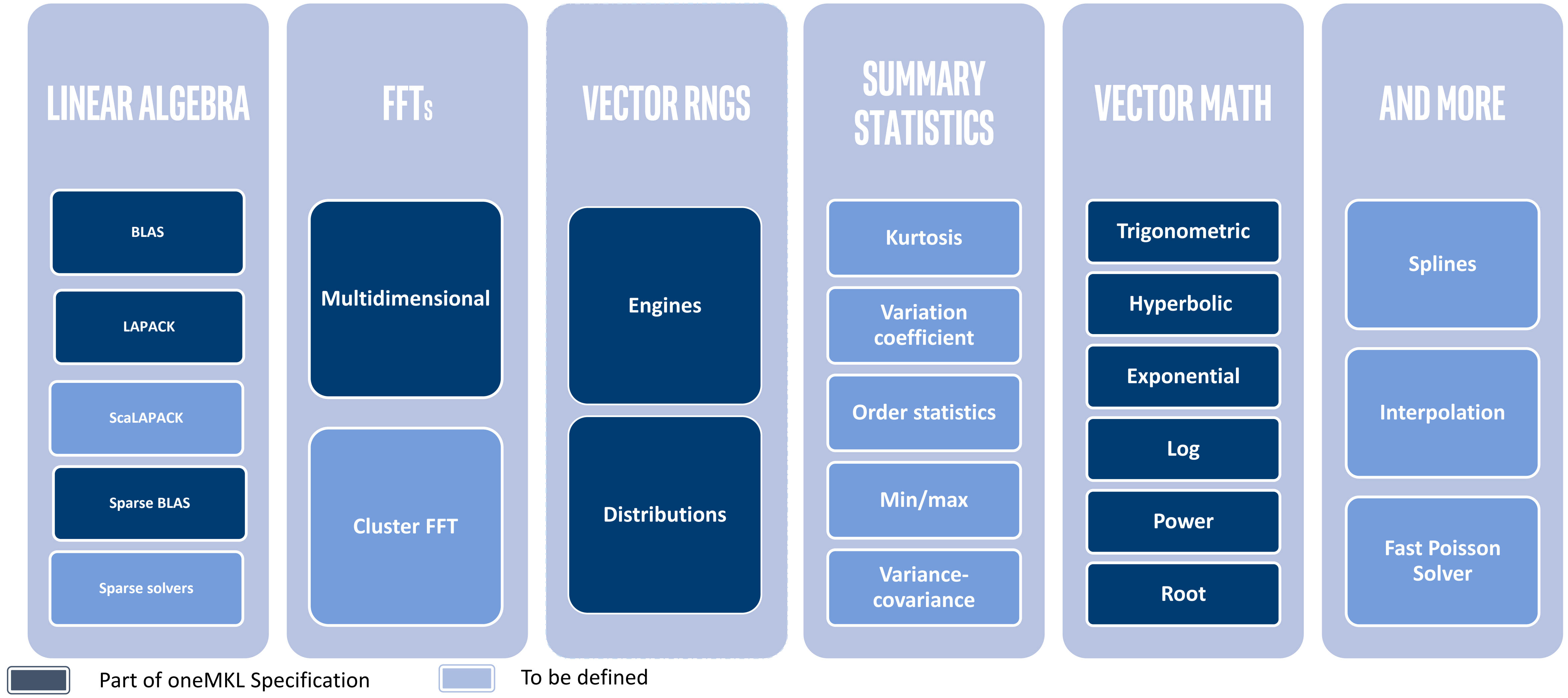
**USM API**

```
using onemkl::blas::gemm;
int64_t n = 32;
sycl::device dev(sycl::{host,cpu,gpu}_selector());
sycl::queue Q(dev);


size_t bytes = n * n * sizeof(double);
double *A = sycl::malloc_shared(bytes, dev,
                          Q.get_context());
double *B = sycl::malloc_shared(…);
double *C = sycl::malloc_shared(…);
double *D = sycl::malloc_shared(…);

sycl::event e =
gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
               n, n, n, 1.0, A, n, B, n,
               0.0, C, n);

gemm(Q, onemkl::transpose::N,
onemkl::transpose::N,
        n, n, n, 1.0, C, n, A, n,
        0.0, D, n, {e});
```

## Walk-thru of the oneMKL Specification

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

# oneAPI Math Kernel Library Specification



| LINEAR ALGEBRA | FFTs | VECTOR RNGS | SUMMARY STATISTICS | VECTOR MATH | AND MORE |
|---|---|---|---|---|---|
| ■ BLAS | ■ Multidimensional | ■ Engines | □ Kurtosis | ■ Trigonometric | □ Splines |
| ■ LAPACK | □ Cluster FFT | ■ Distributions | □ Variation coefficient | ■ Hyperbolic | □ Interpolation |
| □ ScaLAPACK | | | □ Order statistics | ■ Exponential | □ Fast Poisson Solver |
| ■ Sparse BLAS | | | □ Min/max | ■ Log | |
| □ Sparse solvers | | | □ Variance-covariance | ■ Power | |
| | | | | ■ Root | |

■ Part of oneMKL Specification □ To be defined

## 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):
  - Overview of the open source oneMKL interfaces GitHub project
  - Particular feedback requests on oneMKL Spec:
    - Asynchronous execution
    - Multi GPU execution
    - Exceptions/error codes

## 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