SPEED UP NUMERICAL APPLICATION PERFORMANCE WITH INTEL® ONEAPI MATH KERNEL LIBRARY PUBLIC BETA

Fastest and Most Used Math Library for Intel®-based Systems[1]

Core Performance & Developer Products (CPDP) Division
Intel Architecture, Graphics & Software (IAGS)

[1]Data from Evans Data Software Developer surveys, 2011-2019

INTRODUCING ONEAPI

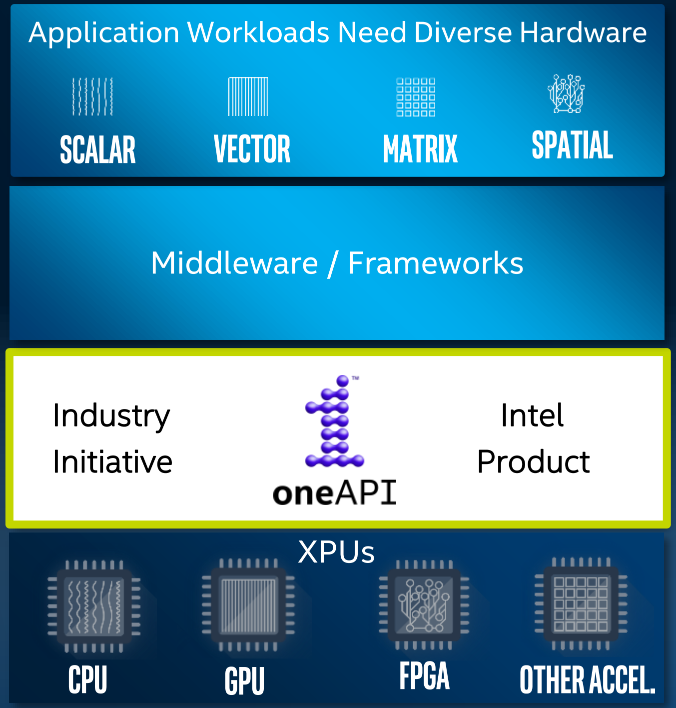
Unified programming model to simplify development across diverse architectures

Unified and simplified language and libraries for expressing parallelism

Uncompromised native high-level language performance

Based on industry standards and open specifications

Interoperable with existing HPC programming models



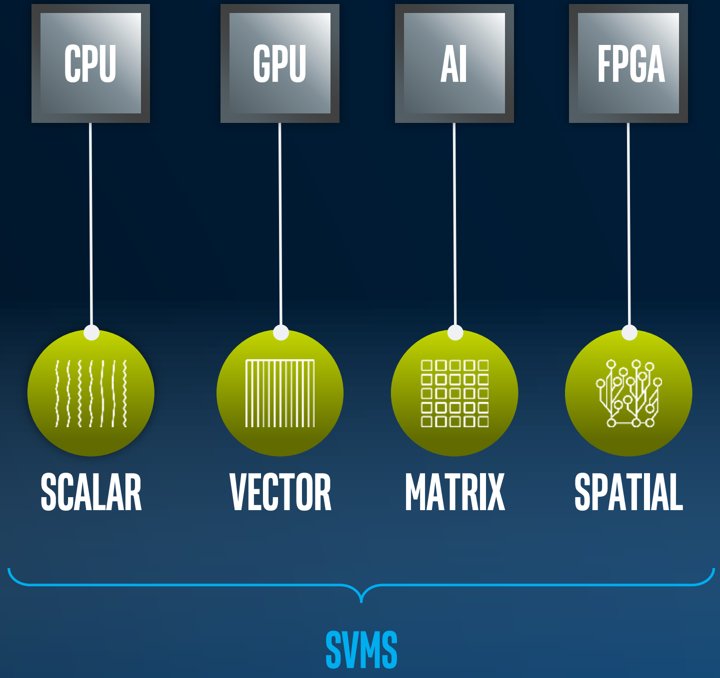
BENEFITS OF ONEAPI PROGRAMMING

Support for a diverse set of data-centric hardware

Common programming language or APIs

Consistent tool support across platforms

Shared software investment across platforms



INTEL® oneAPI MATH KERNEL LIBRARY BETA

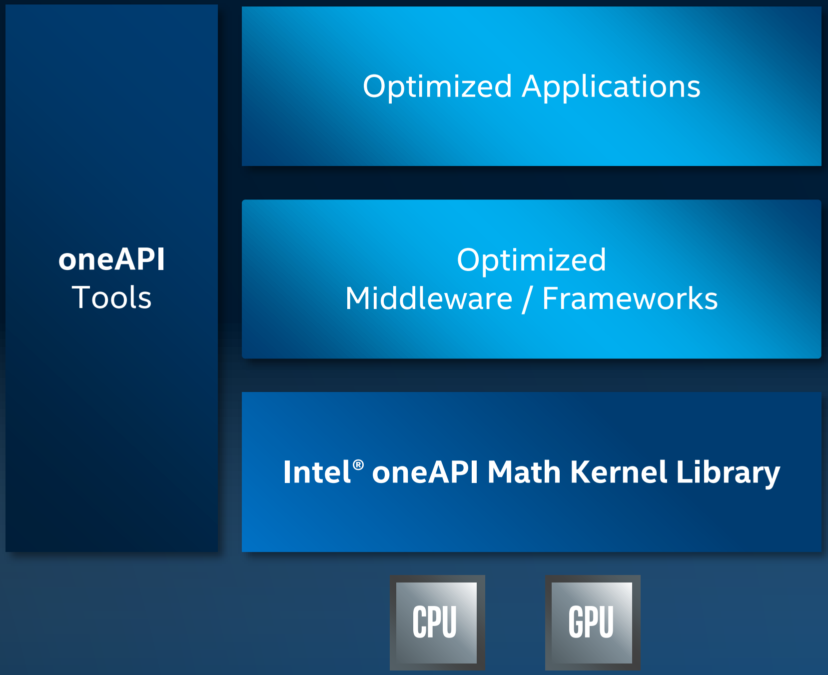
Common developer experience across Scalar, Vector, Matrix and Spatial (SVMS) architecture

Unified and simplified language and libraries for expressing parallelism

Uncompromised native high-level language performance

Support for CPU & GPU

Based on industry standards and open APIs



Intel® oneAPI Math Kernel Library(Beta)

Available Q4 2019

- Language support for Intel® Data Parallel C++/C/Fortran
- Available at no cost and royalty free
- Great performance with minimal effort
- Full support for CPUs, and select support for Intel® Processor Graphics Gen9

▪ Speeds computations for scientific, engineering, financial applications by providing highly optimized, threaded, and vectorized math functions

▪ Provides key functionality for dense and sparse linear algebra (BLAS, LAPACK, PARDISO), FFTs, vector math, summary statistics, splines and more

▪ Dispatches optimized code for each processor automatically without the need to branch code

▪ Optimized for single core vectorization and cache utilization

▪ Automatic parallelism for multi-core CPUs, GPUs and scales from core to clusters

INTEL® ONEAPI MATH KERNEL LIBRARY OFFERS...

- DENSE LINEAR ALGEBRA
- SPARSE LINEAR ALGEBRA
- FAST FOURIER TRANSFORMS
- VECTOR MATH
- VECTOR RNGS
- AND MORE!

Available ONLY as a part of Intel®oneAPI Base Toolkit

What's Inside Intel® oneAPI Math Kernel Library^{Beta} (oneMKL)

LINEAR ALGEBRA
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- Cluster Sparse Solver
- PARDISO*

FFTs
- Multidimensional
- FFTW interfaces
- Cluster FFT

VECTOR RNGS
- Engines
- Distributions

SUMMARY STATISTICS
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

VECTOR MATH
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

AND MORE
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Beta Intel® Processor Graphics Gen9 support

Limited - Beta Intel® Processor Graphics Gen9 support (see release notes)

CPU C/Fortran support

Refer to software.intel.com/articles/optimization-notice for more information regarding performance & optimization choices in Intel software products.

What's New for Intel® oneAPI Math Kernel Library^{Beta} (oneMKL)

- Support for Intel® Processor Graphics Gen9
- Support for Intel® Data Parallel C++ language bindings:
  - BLAS – full support CPU & GPU
  - LAPACK – CPU: Select dense linear solvers, select dense eigensolvers and select batched LAPACK functions; GPU: Batched & non-batched: LU factorization/solve/inverse, Cholesky factorization/solve, QR factorization; Non-batched: triangular matrix solve, symmetric eigensolver
  - FFT – CPU: 1D, 2D, 3D, C2C; GPU: 1D, C2C
  - VS – CPU: full support of pseudo-random and quasi-random Engines, continues (except gaussian_mv) and Discrete Distributions; GPU: Philox4x32-10 and Mrg32k3a Engines, Uniform/Gaussian/Log-normal/Discrete Uniform/Unifrom Bits distributions.
  - VM – CPU & GPU
- Limited support for openMP variant of Intel® Processor Graphics Gen9 offload for C/C++
  - BLAS - S/GEMM

DPC++ API Example General Matrix Multiply (GEMM)

Intel® oneAPI Math Kernel Library(Beta)

DPC++ API

```
using namespace cl::sycl;

int64_t m = 10, n = 6, k = 8,
    lda = 12, ldb = 8, ldc = 10;
double alpha = 1.0, beta = 0.0;

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

// Prepare buffers
buffer<double, 1> A_buf (A, range<1>(lda * k));
buffer<double, 1> B_buf (B, range<1>(ldb * n));
buffer<double, 1> C_buf (C, range<1>(ldc * n));

// Set up device and queue
device dev({host, cpu, gpu}_selector());
queue Q(dev);

// Compute C = A * B
mkl::blas::gemm(Q,mkl::transpose::N,mkl::transpose::N,
    m, n, k, alpha, A_buf, lda, B_buf, ldb,
    beta, C_buf, ldc);
```

sizes and scalars

host memory

DPC++ buffers

device can be host, cpu or gpu

C = A * B

CBLAS API

```
MKL_INT m = 10, n = 6, k = 8,
    lda = 12, ldb = 8, ldc = 10;
double alpha = 1.0, beta = 0.0;

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

// Compute C = A * B
cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans,
    m, n, k, alpha, A, lda, B, ldb, beta, C, ldc);
```

OpenMP Offload Example

Intel® oneAPI Math Kernel Library(Beta)

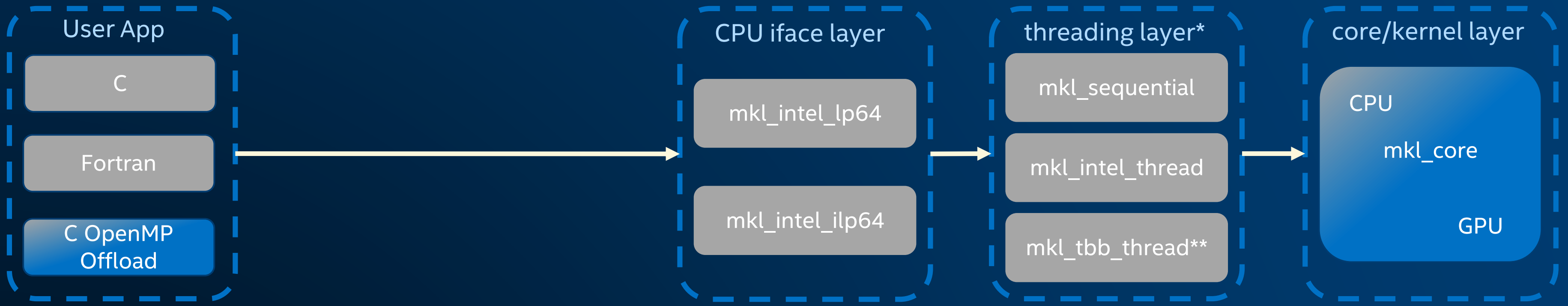
```
#pragma omp target data map(to:a[0:sizea],b[0:sizeb]) map(tofrom:c[0:sizec]) device(dnum)
{

// Call a standard CBLAS function
#pragma omp target variant dispatch device(dnum) use_device_ptr(a, b, c)
    {
        cblas_sgemm(CblasColMajor, CblasNoTrans, CblasNoTrans, m, n, k, alpha, a, lda, b, ldb, beta, c, ldc);
    }
}
```

Executes even when offload does not occur

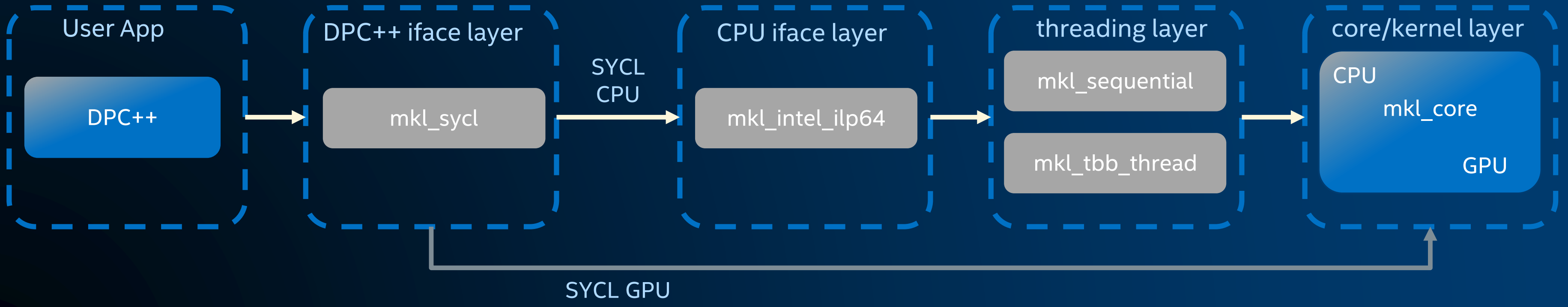
- cblas_sgemm() is a standard CBLAS function available from Intel MKL
- This is valid host-side code even if offload is disabled

Intel® oneAPI Math Kernel(Beta) Library Offload Configuration

CPU
GPU



* C/Fortran applications also support PGI and GNU OpenMP threading layers
** C OpenMP offload does not support TBB threading layer



NOTICES & DISCLAIMERS

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.
Notice revision #20110804

BACKUP

Intel® Math Kernel Library BLAS

(Basic Linear Algebra Subprograms)

De-facto Standard APIs since the 1980s	
100s of Basic Linear Algebra Functions	Level 1 – vector vector operations, O(N) Level 2 – matrix vector operations, $O(N^2)$ Level 3 – matrix matrix operations, $O(N^3)$
Precisions Available	Real – Single and Double Complex - Single and Double
BLAS-like Extensions	Direct Call, Batched, Packed and Compact
Reference Implementation	*http://netlib.org/blas/*

Intel® Math Kernel Library LAPACK

(Linear Algebra PACKage)

De-facto Standard APIs since the 1990s	
1000s of Linear Algebra Functions	Matrix factorizations - LU, Cholesky, QR Solving systems of linear equations Condition number estimates Symmetric and non-symmetric eigenvalue problems Singular value decomposition and many more...
Precisions Available	Real – Single and Double, Complex – Single and Double
Reference Implementation	*http://netlib.org/lapack/*

Intel® Math Kernel Library Fast Fourier Transforms (FFTs)

FFTW Interfaces support	C, C++ and FORTRAN source code wrappers provided for FFTW2 and FFTW3. FFTW3 wrappers are already built into the library
Cluster FFT	Perform Fast Fourier Transforms on a cluster Interface similar to DFTI Multiple MPIs supported
Parallelization	Thread safe with automatic thread selection
Storage Formats	Multiple storage formats such as CCS, PACK and Perm
Batch support	Perform multiple transforms in a single call
Additional Features	Perform FFTs on partial images Padding added for better performance Transform combined with transposition Mixed-language usage supported

Intel® Math Kernel Library Vector Math

Example:	y(i) = $e^{x(i)}$ for *i =1 to n*
Broad Function Support	Basic Operations – add, sub, mult, div, sqrt Trigonometric– sin, cos, tan, asin, acos, atan Exponential – exp,, pow, log, log10, log2, Hyperbolic – sinh, cosh, tanh Rounding – ceil, floor, round And many more
Precisions Available	Real – Single and Double Complex - Single and Double
Accuracy Modes	High - almost correctly rounded Low - last 2 bits in error Enhanced Performance - 1/2 the bits correct

Intel® Math Kernel Library Vector Statistics

Random Number Generators (RNGs)	Pseudorandom, quasi-random and non-deterministic random number generators with continuous and discrete distribution
Summary Statistics	Parallelized algorithms to compute basic statistical estimates for single and double precision multi-dimensional datasets
Convolution & Correlation	Linear convolution and correlation transforms for single and double precision real and complex data

Intel® Math Kernel Library Sparse Solvers

PARDISO - Parallel Direct Sparse Solver	Factor and solve Ax = b using a parallel shared memory *LU*, *LDL*, or LL^T factorization Supports a wide variety of matrix types including real, complex, symmetric, indefinite, ... Includes out-of-core support for very large matrix sizes
Parallel Direct Sparse Solver for Clusters	Factor and solve Ax = b using a parallel distributed memory *LU*, *LDL*, or LL^T factorization Supports a wide variety of matrix types (real, complex, symmetric, indefinite, ...) Supports A stored in 3-array CSR3 or BCSR3 formats
DSS – Simplified PARDISO Interface	An alternative, simplified interface to PARDISO
ISS – Iterative Sparse Solvers	Conjugate Gradient (CG) solver for symmetric positive definite systems Generalized Minimal Residual (GMRes) for non-symmetric indefinite systems Rely on Reverse Communication Interface (RCI) for matrix vector multiply

Intel® Math Kernel Library General Components

Sparse BLAS	NIST-like and inspector execute interfaces
Data Fitting	1D linear, quadratic, cubic, step-wise and user-defined splines, spline-based interpolation and extrapolation
Partial Differential Equations	Helmholtz, Poisson, and Laplace equations
Optimization	Trust-region solvers for nonlinear least square problems with and without constraints
Service Functions	Threading controls Memory management Numerical reproducibility

Intel® Math Kernel Library Summary

Boosts application performance with minimal effort	feature set is robust and growing provides scaling from the core, to multicore, to manycore, and to clusters automatic dispatching matches the executed code to the underlying processor future processor optimizations included well before processors ship
Showcases the world's fastest supercomputers[1]	Intel® Distribution for LINPACK* Benchmark Intel® Optimized High Performance Conjugate Gradient Benchmark

[1]Data from Evans Data Software Developer surveys, 2011-2019

Refer to software.intel.com/articles/optimization-notice for more information regarding performance & optimization choices in Intel software products.

Intel® Math Kernel Library Resources

Intel® Math Kernel Library (Intel® MKL) site Intel® oneAPI Math Kernel Library(Beta)	software.intel.com/intel-mkl software.intel.com/oneAPI/mkl
Intel® MKL Forum	software.intel.com/forums/intel-math-kernel-library
Intel® MKL Benchmarks	software.intel.com/intel-mkl/benchmarks#
Intel® MKL Link Line Advisor	software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Software