Intel® oneAPI Math Kernel Library (oneMKL)

INTEL / LRZ oneAPI Workshop, June 5th-7th, 2023

Gennady.Fedorov@intel.com

Intel® oneAPI Base Toolkit

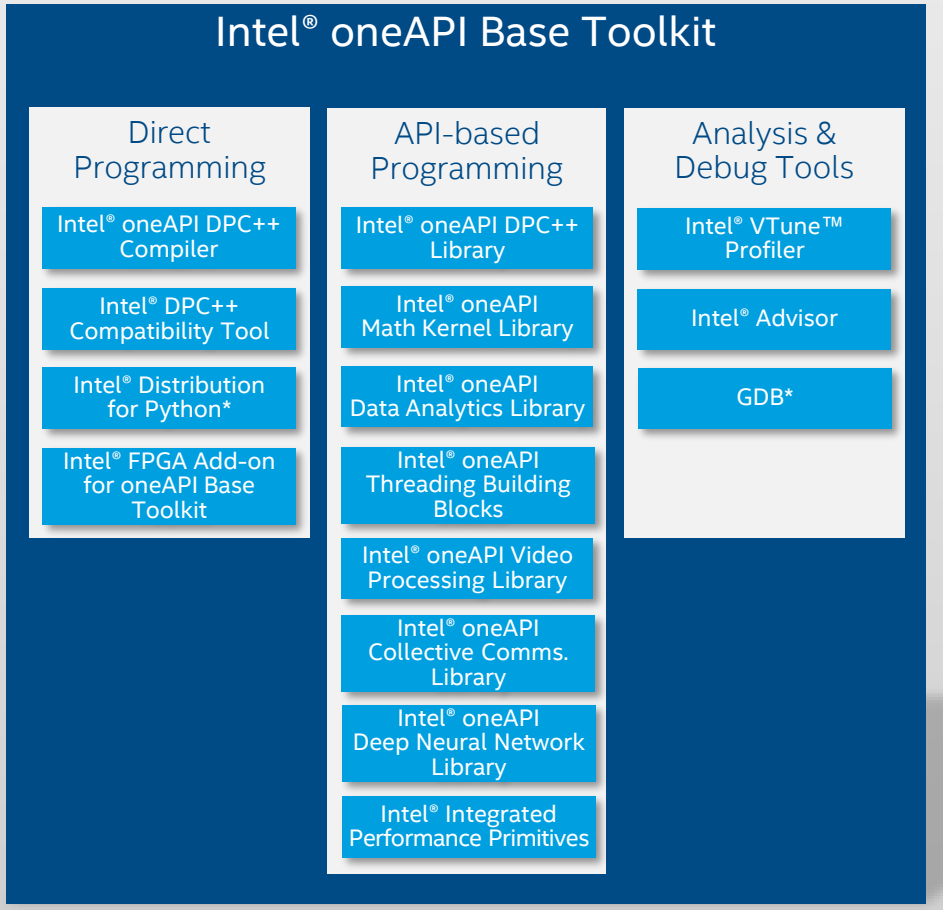
Core set of frequently used tools and libraries for developing high-performance applications across diverse architectures—CPU, GPU, FPGA.

Who Uses It?

- A broad range of developers across industries
- Add-on toolkit users because this is the base for all toolkits

Top Features/Benefits

- Data Parallel C++ (DPC++) compiler, library, and analysis tools
- DPC++ Compatibility tool helps migrate existing CUDA code
- Python distribution includes accelerated scikit-learn, NumPy, SciPy libraries
- Optimized performance libraries for threading, math, data analytics, deep learning, and video/image/signal processing



Intel® oneAPI Base Toolkit

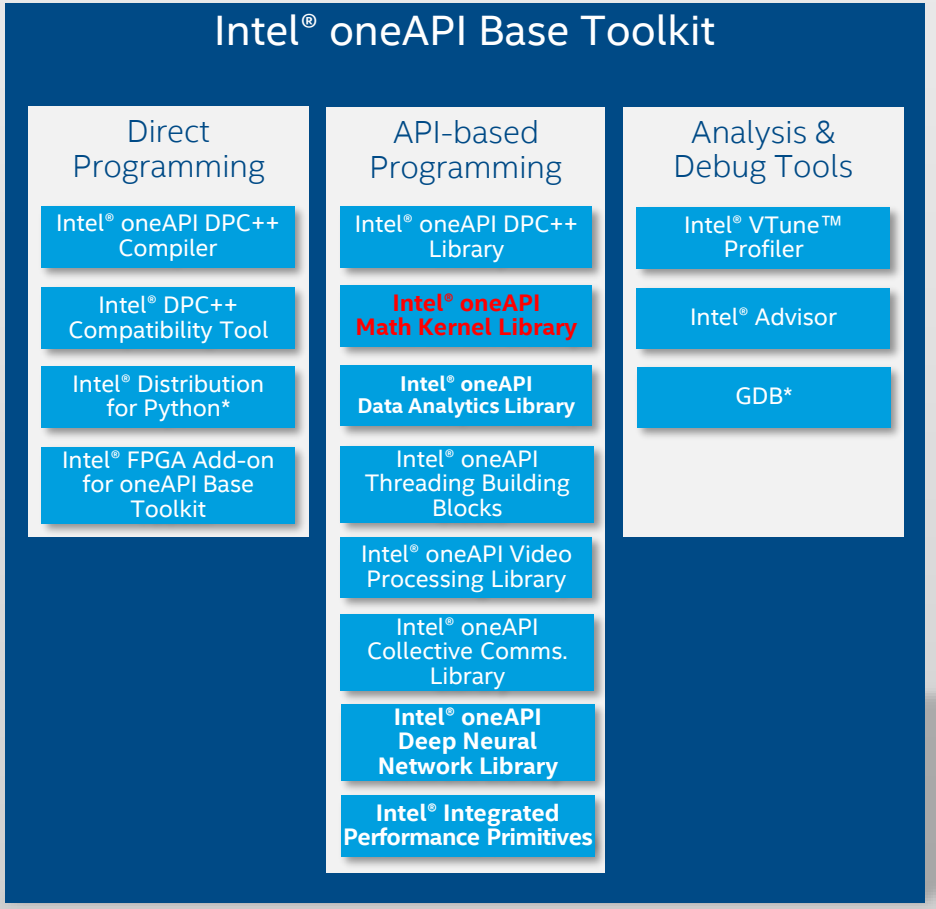
Core set of frequently used tools and libraries for developing high-performance applications across diverse architectures—CPU, GPU, FPGA.

Who Uses It?

- A broad range of developers across industries
- Add-on toolkit users because this is the base for all toolkits

Top Features/Benefits

- Data Parallel C++ (DPC++) compiler, library, and analysis tools
- DPC++ Compatibility tool helps migrate existing CUDA code
- Python distribution includes accelerated scikit-learn, NumPy, SciPy libraries
- Optimized performance libraries for threading, math, data analytics, deep learning, and video/image/signal processing

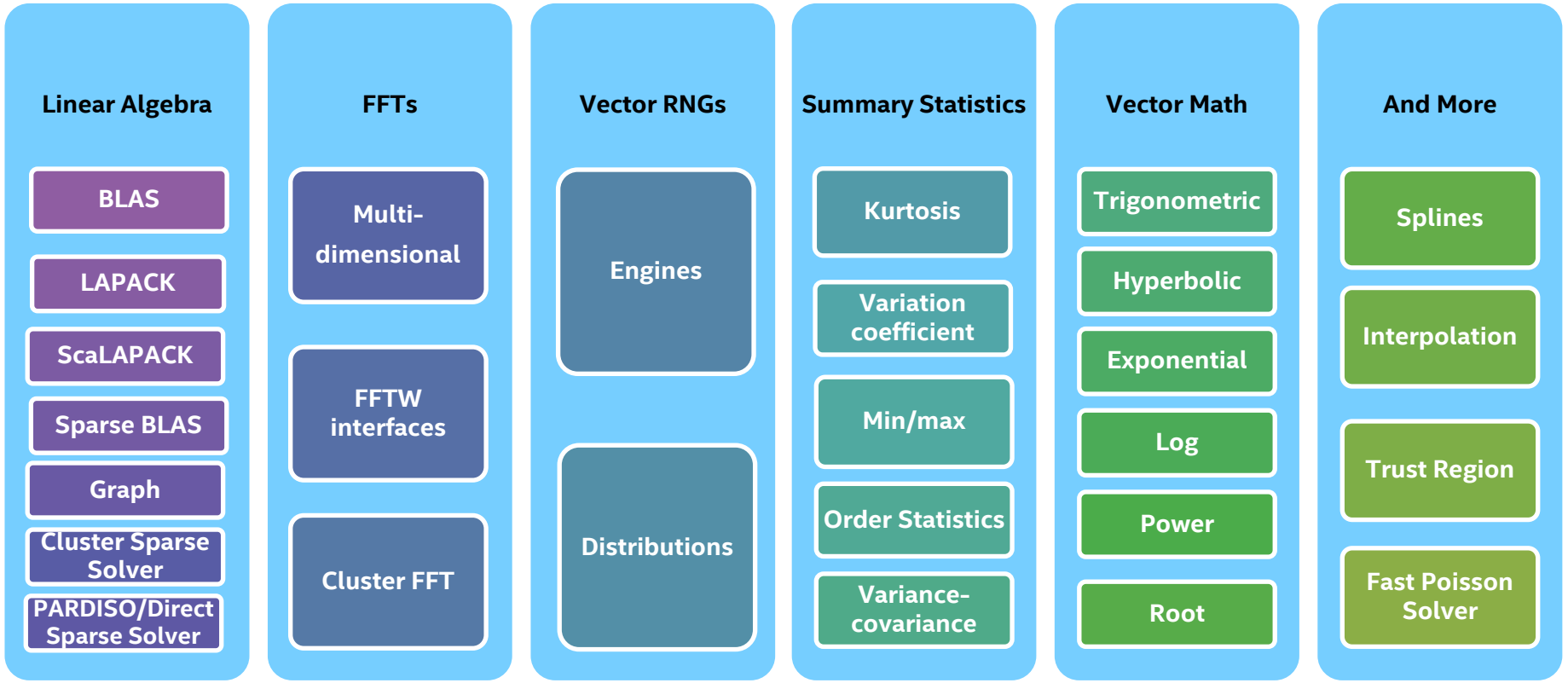


oneMKL, 32 bit – **Your Feedback Matters**

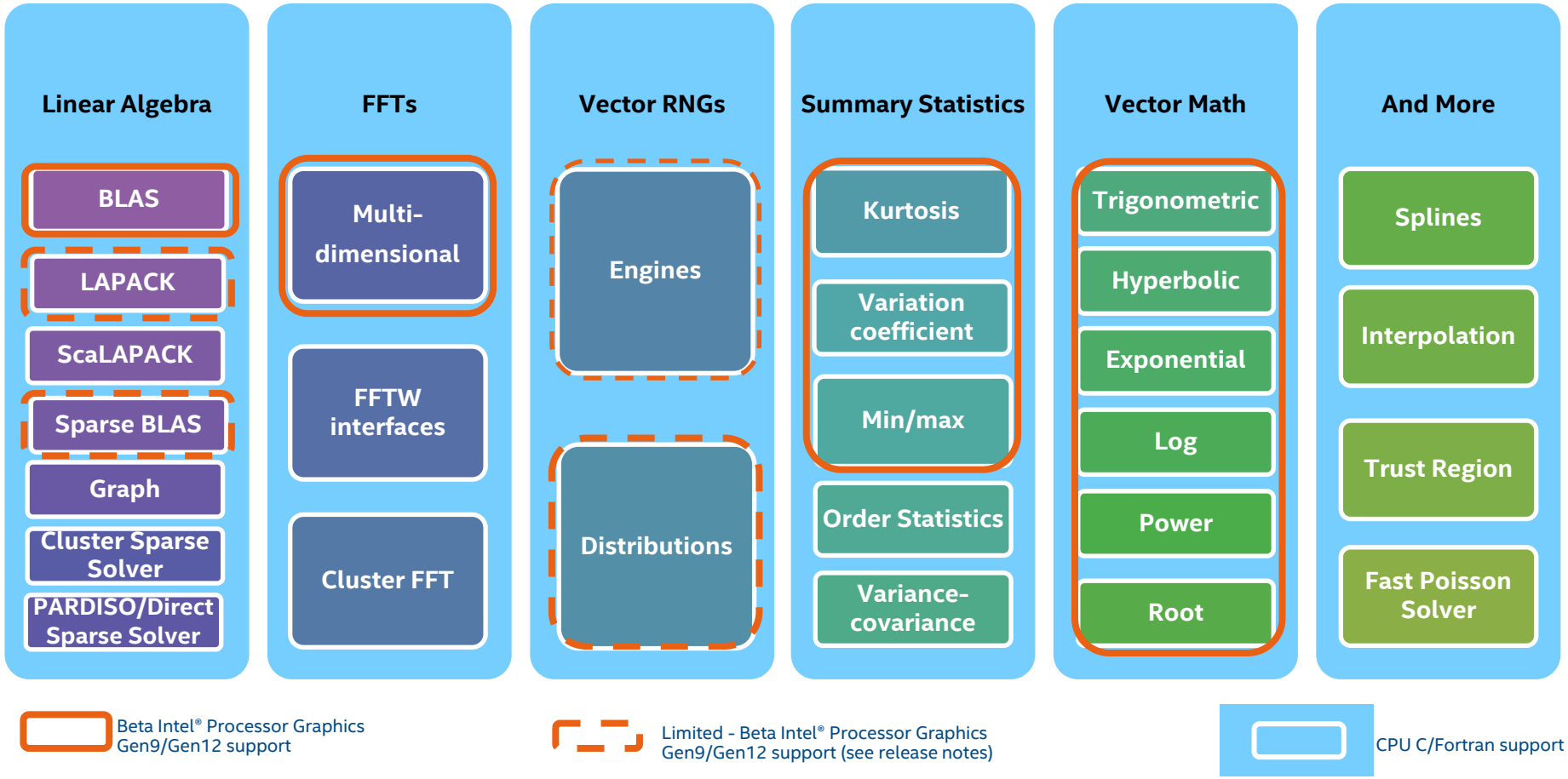
1	Are you currently using Intel® oneAPI Math Kernel Library (Intel® oneMKL) 32-bit library?	Yes / No
2	If you answered yes, what are some of the applications you're using Intel® oneMKL 32-bit library for?	Please enter use cases / applications
3	When do you plan to move to 64-bit library?	0 - 6 months
		6 months – 1 year
		1 – 2 years
		No current plans to move

https://community.intel.com/t5/Intel-oneAPI-Math-Kernel-Library/Your-Feedback-Matters/m-p/1484337/emcs_t/S2h8ZW1haWx8Ym9hcmRfc3Vic2NyaXB0aW9ufExIRko5M1U3QTAyTklLfDE0ODQzMzd8U1VCU0NSSVBUSU9OU3xoSw#M34561

Intel® oneAPI Math Kernel Library (oneMKL)

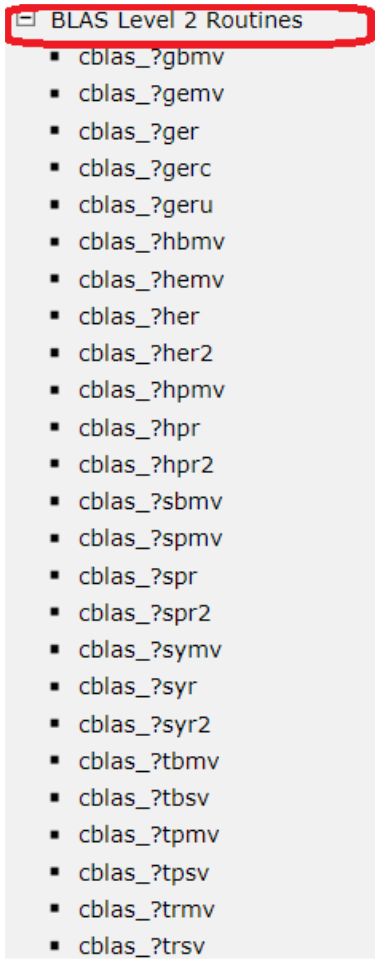


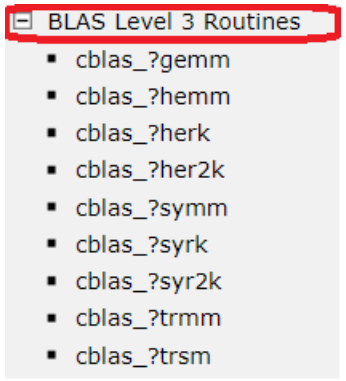
Intel® oneAPI Math Kernel Library (oneMKL), cont.



Intel® oneAPI MKL, BLAS, update

- BLAS, Netlib interfaces
  - USM and Buffer API - ALL
  - C/Fortran Offloading – ALL
  - BLAS Extensions





OpenMP offload to support the OpenMP* 5.1 specification

Intel® oneAPI MKL, BLAS, update, cont.

- BLAS, Netlib interfaces
  - USM and Buffer API - ALL
  - C/Fortran Offloading – ALL
  - **BLAS Extensions**

CPU	OpenMP Offload Intel GPU
{AXPY,GEMM,TRSM}_BATCH (group and strided)	{AXPY,GEMM,TRSM}_BATCH (group and strided)
GEMMT, AXPBY, GEMM3M	GEMMT
Integer GEMM (s8u8)	N/A
Bfloat16 GEMM	N/A
JIT GEMM API	N/A
PACK GEMM API	N/A
COMPACT GEMM API	N/A

oneMKL, DGEMM, C API

$$C \leftarrow \alpha AB + \beta C$$

```
int main() {

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


    // Allocate matrices
    double *A = (double *) mkl_malloc(sizeof(double) * sizea);
    double *B = (double *) mkl_malloc(sizeof(double) * sizeb);
    double *C = (double *) mkl_malloc(sizeof(double) * sizec);

    // Initialize matrices [...]
    ...
    cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans, m, n, k,
            alpha, A, lda, B, ldb, beta, C, ldc);
    ...
}
```

oneMKL, DGEMM, DPCPP API

$$C \leftarrow \alpha AB + \beta C$$

```
using namespace oneapi::mkl::blas
int main(....) {

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

   sycl::queue Q{sycl::gpu_selector{}};

   // Allocate matrices
   double *A = malloc_shared<double>(sizea, Q);
   double *B = malloc_shared<double>(sizeb, Q);
   double *C = malloc_shared<double>(sizec, Q);

   // Initialize matrices [...]
   ...

   auto e =:gemm(queue, transpose::N, transpose::N, m, n, k,

                        alpha, A, lda, B, ldb, beta, C, ldc));

   ...
}
```

Set up GPU queue

Allocate CPU/GPU-accessible shared memory

New oneMKL DPC++ API

Computation is performed on given queue

Output **e** is a sycl::event object representing command completion

Call **`e.wait()`** to wait for completion

What's New for Intel® oneAPI Math Kernel Library (oneMKL) 2023.0

oneMKL on 4th Gen Intel® Xeon® Scalable processors

Maximize performance with oneMKL, unleashing the power of built-in accelerators

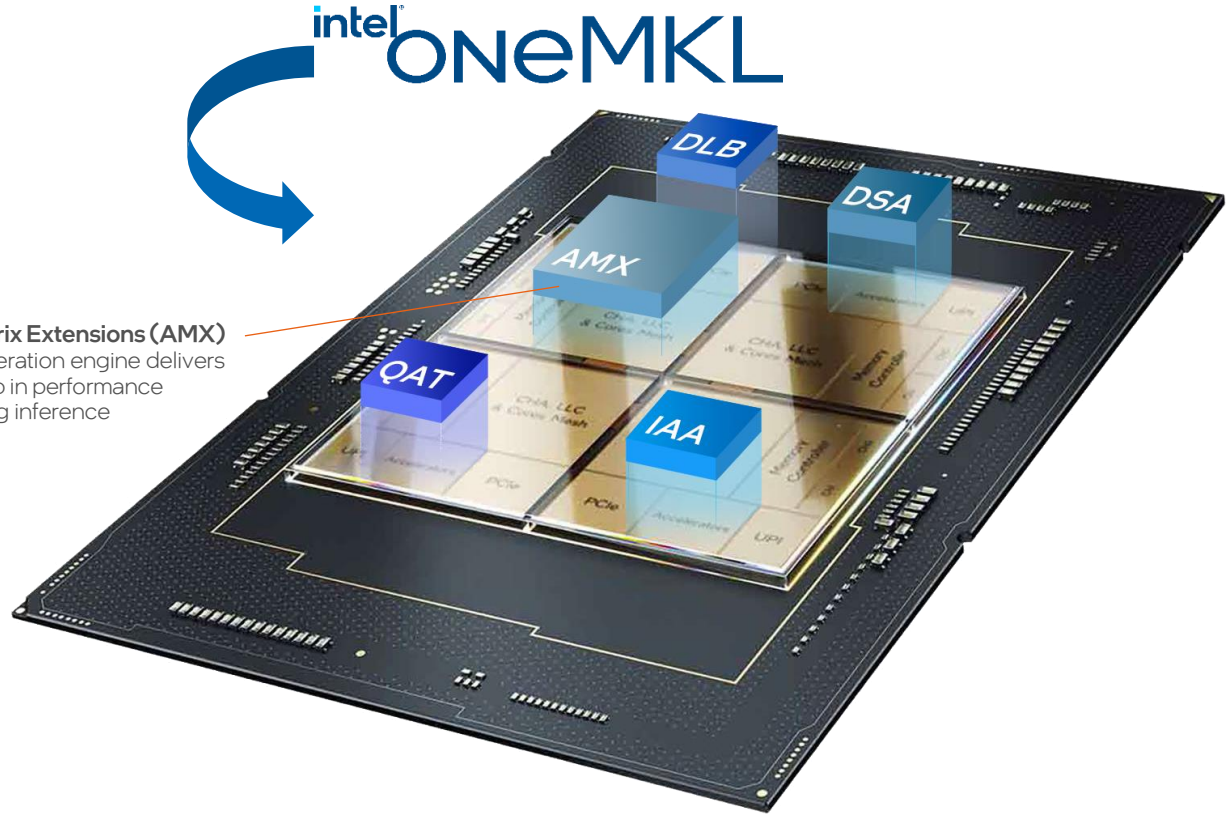
- The Intel® oneAPI Math Kernel Library (oneMKL) leverages Intel® AMX-Advanced Matrix eXtensions to optimize matrix computations for the BF16 and INT8 data types.
- oneMKL also leverages Intel ® AVX-512 -Advanced Vector Extensions for the FP16 data type on 4th Gen Intel® Xeon® Scalable processors.
- Most oneMKL memory- bound dense and sparse linear algebra (BLAS, LAPACK, sparse direct solvers), FFT, vector math, vector RNG, summary statistics, or spline computations, directly benefit from the onboard High Bandwidth Memory (HBM).

intel XEON SILVER

intel XEON GOLD

intel XEON PLATINUM

Advanced Matrix Extensions (AMX)
Built-in AI acceleration engine delivers a significant leap in performance for deep learning inference and training.

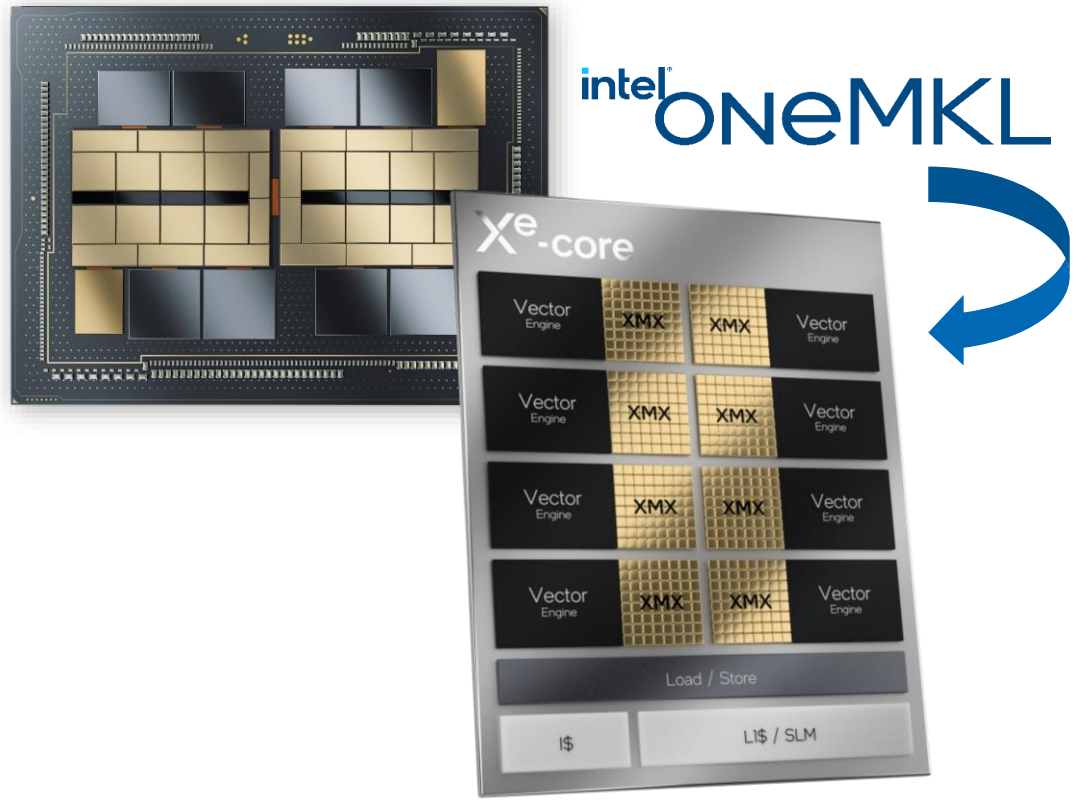


4th Gen Intel® Xeon® Scalable Processors with Intel® Advanced Matrix Extensions, Quick assist Technology, Intel® AVX-512, bfloat16, and more built-in accelerators

oneMKL on Intel® Data Center GPU Max Series

Breakthrough Performance for HPC and AI

- The Intel® oneAPI Math Kernel Library (oneMKL) leverages Intel® Xe Matrix Extensions (Intel® XMX) to optimize matrix computations for TF32, FP16, BF16 and INT8 data types on Intel® Data Center GPU Max Series (codenamed Ponte Vecchio).
- oneMKL provides a variety of dense and sparse linear algebra (BLAS, LAPACK, sparse BLAS), FFT, vector math, vector RNG, summary statistics, and spline interfaces both for the SYCL and C/Fortran OpenMP* offload programming models to enable applications targeting Intel® Data Center GPUs.



Intel® Data Center GPUs with hardware AV1 encode and Max with datatype flexibility, Intel® Xe Matrix Extensions, vector engine, XE-Link, and other features

oneAPI Math Kernel Library (oneMKL) Interfaces

- oneAPI specification

  https://spec.oneapi.io/versions/latest/introduction.html

- oneMKL specification
  https://spec.oneapi.io/versions/latest/elements/oneMKL/source/index.html

- One MKL Open-Source interface

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

oneAPI MKL Interfaces Project

			Type	Compiler
BLAS	x86 CPU	Intel(R) oneAPI Math Kernel Library	Dynamic, Static	DPC++, LLVM*, hipSYCL
	Intel GPU		Dynamic, Static	DPC++
	NVIDIA GPU	NVIDIA cuBLAS	Dynamic, Static	LLVM*, hipSYCL
	x86 CPU	NETLIB LAPACK	Dynamic, Static	DPC++, LLVM*, hipSYCL
	AMD GPU	AMD rocBLAS	Dynamic, Static	LLVM*, hipSYCL
	x86 CPU, Intel GPU, NVIDIA GPU, AMD GPU	SYCL-BLAS	Dynamic, Static	DPC++, LLVM*
LAPACK	x86 CPU	Intel(R) oneAPI Math Kernel Library	Dynamic, Static	DPC++, LLVM*
	Intel GPU		Dynamic, Static	DPC++
	NVIDIA GPU	NVIDIA cuSOLVER	Dynamic, Static	LLVM*
	AMD GPU	AMD rocSOLVER	Dynamic, Static	LLVM*
RNG	x86 CPU	Intel(R) oneAPI Math Kernel Library	Dynamic, Static	DPC++, LLVM*, hipSYCL
	Intel GPU		Dynamic, Static	DPC++
	NVIDIA GPU	NVIDIA cuRAND	Dynamic, Static	LLVM*, hipSYCL
	AMD GPU	AMD rocRAND	Dynamic, Static	LLVM*, hipSYCL
DFT	Intel GPU	Intel(R) oneAPI Math Kernel Library	Dynamic, Static	DPC++
	x86 CPU		Dynamic, Static	DPC++
	NVIDIA GPU	NVIDIA cuFFT	Dynamic, Static	DPC++

oneMKL, BLAS, GPU, the new compute modes

- The new MKL_BLAS_COMPUTE_MODE is intended for quickly evaluating whether alternate compute modes provide performance benefits and acceptable accuracy for an application. After initial testing, alternate mode settings can be permanently applied within the application using the per-call or per-source-file APIs.
- New alternate computation mode functionality for Level-3 routines
  - Better performance
  - Reduced accuracy
  - oneMKL does not enable any alternate compute modes by the default
  - The same or optional interfaces
  - Limitations: gemm, gemmt, syrk and syrk2 (MKL 2023)

oneMKL, BLAS, GPU, the new compute modes, cont.

```
from blas.hpp
#define ONEMKL_DECLARE_GEMM(Ta, Tb, Tc, Ts) \
DLL_EXPORT sycl::event gemm(sycl::queue &queue, transpose transa, transpose transb, \
                            std::int64_t m, std::int64_t n, std::int64_t k, \
                            Ts alpha, const Ta *a, std::int64_t lda, \
                            const Tb *b, std::int64_t ldb, \
                            Ts beta, Tc *c, std::int64_t ldc, \
                            compute_mode mode, 
                            const std::vector<sycl::event> &dependencies = {}); \
ONEMKL_INLINE_DECLARE sycl::event gemm(sycl::queue &queue, transpose transa, transpose transb, \
                              std::int64_t m, std::int64_t n, std::int64_t k, \
                              Ts alpha, const Ta *a, std::int64_t lda, \
                              const Tb *b, std::int64_t ldb, \
                              Ts beta, Tc *c, std::int64_t ldc, \
                              const std::vector<sycl::event> &dependencies = {}) \
{ \
   return gemm(queue, transa, transb, m, n, k, alpha, a, lda, b, ldb, beta, c, ldc, MKL_BLAS_COMPUTE_MODE, dependencies); \
}
```

BLAS_64/Lapack_64 API Extensions

- Using BLAS and LAPACK with the 32-bit and 64-bit interface (lp64 / ilp64) at the same time
- BLAS_64 and LAPACK_64 - NetLib interfaces
- Declaration: mkl_blas_64.h, mkl_lapack.h
- Limitations :
  - Intel64 only.
  - no Fortran API at this moment
  - no mkl_lapacke.h (LAPACKE_cgetrf(*.....))
  - CPU only

BLAS_64/Lapack_64 API Extensions, cont.

- BLAS:
  - void sgemm(const char *transa, const char *transb, const MKL_INT *m, const MKL_INT *n, const MKL_INT *k, const float *alpha, const float *a, const MKL_INT *lda, const float *b, const MKL_INT *ldb, const float *beta, float *c, const MKL_INT *ldc)
  - void sgemm_64(const char *trans, const MKL_INT64 *m, const MKL_INT64 *n, const float *alpha, const float *a, const MKL_INT64 *lda, const float *x, const MKL_INT64 *incx, const float *beta, float *y, const MKL_INT64 *incy)
- LAPACK:
  - void sgetrf(const MKL_INT* m, const MKL_INT* n, float* a, const MKL_INT* lda, MKL_INT* ipiv, MKL_INT* info)
  - void sgetrf_64(const MKL_INT64* m, const MKL_INT64 * n, float* a, const MKL_INT64 * lda, MKL_INT64 * ipiv, MKL_INT64 * info)

Demo, MKL

Intel® oneMKL Resources

Intel® oneMKL Product Page	https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
Get Started with Intel® oneMKL	https://www.intel.com/content/www/us/en/develop/documentation/get-started-with-mkl-for-dpcpp/top.html
Intel® oneMKL Developer Reference	https://www.intel.com/content/www/us/en/develop/documentation/onemkl-developer-reference-c/top.html
Intel® oneMKL Developer Guide	https://www.intel.com/content/www/us/en/develop/documentation/onemkl-windows-developer-guide/top.html
Intel® oneMKL Specification	https://spec.oneapi.io/versions/latest/elements/oneMKL/source/index.html
Intel® oneMKL Open-Source Interfac	https://github.com/oneapi-src/oneMKL
Intel® oneMKL Release Notes	https://cqpreview.intel.com/content/www/us/en/developer/articles/release-notes/onemkl-release-notes.html
Intel® oneMKL Forum	https://community.intel.com/t5/Intel-oneAPI-Math-Kernel-Library/bd-p/oneapi-math-kernel-library

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at intel.com or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

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. https://software.intel.com/en-us/articles/optimization-notice

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. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.


intel