# Intel MKL

## Intel Math Kernel Library

Intel Math Kernel Library (Intel MKL) is a library of math kernel subroutines, extensively threaded and optimized for maximum performance. Intel MKL provides these basic math kernels:

- BLAS (level 1, 2, and 3) and LAPACK linear algebra routines, offering vector, vector-matrix, and matrix-matrix operations.
- The PARDISO direct sparse solver, an iterative sparse solver, and supporting sparse BLAS (level 1, 2, and 3) routines for solving sparse systems of equations.
- ScaLAPACK distributed processing linear algebra routines for Linux* and Windows* operating systems, as well as the Basic Linear Algebra Communications Subprograms (BLACS) and the Parallel Basic Linear Algebra Subprograms (PBLAS).
- Fast Fourier transform (FFT) functions in one, two, or three dimensions with support for mixed radices (not limited to sizes that are powers of 2), as well as distributed versions of these functions.
- Vector Math Library (VML) routines for optimized mathematical operations on vectors.
- Vector Statistical Library (VSL) routines, which offer high-performance vectorized random number generators (RNG) for several probability distributions, convolution and correlation routines, and summary statistics functions.
- Data Fitting Library, which provides capabilities for spline-based approximation of functions, derivatives and integrals of functions, and search.
- Extended Eigensolver, a shared memory  version of an eigensolver based on the Feast Eigenvalue Solver.

For details see the Intel MKL Reference Manual.

Intel MKL version 13.5.192 is available on Anselm

```
$ module load mkl
```

The module sets up environment variables, required for linking and running mkl enabled applications. The most important variables are the $MKLROOT, $MKL_INC_DIR, $MKL_LIB_DIR and $MKL_EXAMPLES

The MKL library may be linked using any compiler. With intel compiler use -mkl option to link default threaded MKL.

### Interfaces

The MKL library provides number of interfaces. The fundamental once are the LP64 and ILP64. The Intel MKL ILP64 libraries use the 64-bit integer type (necessary for indexing large arrays, with more than 231^-1 elements), whereas the LP64 libraries index arrays with the 32-bit integer type.

|Interface|Integer type|

—— |—|—|————————————- |LP64|32-bit, int, integer(kind=4), MPI_INT| ILP64 64-bit, long int, integer(kind=8), MPI_INT64

### Linking

Linking MKL libraries may be complex. Intel mkl link line advisor helps. See also examples below.

You will need the mkl module loaded to run the mkl enabled executable. This may be avoided, by compiling library search paths into the executable. Include rpath on the compile line:

```
$ icc .... -Wl,-rpath=$LIBRARY_PATH ...
```

### Threading

Advantage in using the MKL library is that it brings threaded parallelization to applications that are otherwise not parallel.

For this to work, the application must link the threaded MKL library (default). Number and behaviour of MKL threads may be controlled via the OpenMP environment variables, such as OMP_NUM_THREADS and KMP_AFFINITY. MKL_NUM_THREADS takes precedence over OMP_NUM_THREADS

```
$ export OMP_NUM_THREADS=16
$ export KMP_AFFINITY=granularity=fine,compact,1,0
```

The application will run with 16 threads with affinity optimized for fine grain parallelization.

## Examples

Number of examples, demonstrating use of the MKL library and its linking is available on Anselm, in the $MKL_EXAMPLES directory. In the examples below, we demonstrate linking MKL to Intel and GNU compiled program for multi-threaded matrix multiplication.

**Working with examples**

```
$ module load intel
$ module load mkl
$ cp -a $MKL_EXAMPLES/cblas /tmp/
$ cd /tmp/cblas

$ make sointel64 function=cblas_dgemm
```

In this example, we compile, link and run the cblas_dgemm example, demonstrating use of MKL example suite installed on Anselm.

**Example: MKL and Intel compiler**

```
$ module load intel
$ module load mkl
$ cp -a $MKL_EXAMPLES/cblas /tmp/
$ cd /tmp/cblas
$
$ icc -w source/cblas_dgemmx.c source/common_func.c -mkl -o cblas_dgemmx.x
$ ./cblas_dgemmx.x data/cblas_dgemmx.d
```

In this example, we compile, link and run the cblas_dgemm example, demonstrating use of MKL with icc -mkl option. Using the -mkl option is equivalent to:

```
$ icc -w source/cblas_dgemmx.c source/common_func.c -o cblas_dgemmx.x
-I$MKL_INC_DIR -L$MKL_LIB_DIR -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core -liomp5
```

In this example, we compile and link the cblas_dgemm example, using LP64 interface to threaded MKL and Intel OMP threads implementation.

**Example: MKL and GNU compiler**

```
$ module load gcc
$ module load mkl
$ cp -a $MKL_EXAMPLES/cblas /tmp/
$ cd /tmp/cblas

$ gcc -w source/cblas_dgemmx.c source/common_func.c -o cblas_dgemmx.x
-lmkl_intel_lp64 -lmkl_gnu_thread -lmkl_core -lgomp -lm

$ ./cblas_dgemmx.x data/cblas_dgemmx.d
```

In this example, we compile, link and run the cblas_dgemm example, using LP64 interface to threaded MKL and gnu OMP threads implementation.

## MKL and MIC accelerators

The MKL is capable to automatically offload the computations o the MIC accelerator. See section Intel Xeon Phi for details.

## Further reading

Read more on Intel website, in particular the MKL users guide.