# INTEL® MKL V.2019 - BLAS, JIT FEATURE

AUGUST 2018

GENNADY FEDOROV

# CONFIGURATIONS INTEL® PARALLEL STUDIO XE


| Composer Edition | Professional Edition | Cluster Edition |
|---|---|---|
| Intel® Fortran Compiler<br>Intel® C++ Compiler | Intel® Fortran Compiler<br>Intel® C++ Compiler | Intel® Fortran Compiler<br>Intel® C++ Compiler |
| **Intel® Math Kernel Library** | **Intel® Math Kernel Library** | **Intel® Math Kernel Library** |
| Intel® Integrated Performance Primitives<br>Intel® Data Analytics Acceleration Library<br>Intel® Threading Building Blocks<br>® Cilk™ Plus & Intel® OpenMP* | Intel® Integrated Performance Primitives<br>Intel® Data Analytics Acceleration Library<br>Intel® Threading Building Blocks<br>Intel® Cilk™ Plus & Intel® OpenMP* | Intel® Integrated Performance Primitives<br>Intel® Data Analytics Acceleration Library<br>Intel® Threading Building Blocks<br>Intel® Cilk™ Plus & Intel® OpenMP* |
| | Intel® Advisor XE<br>Intel® Inspector XE<br>Intel® VTune™ Amplifier XE | Intel® Advisor XE<br>Intel® Inspector XE<br>Intel® VTune™ Amplifier XE<br>Intel® MPI Library<br>Intel® Trace Analyzer and Collector |
| Bundle or Add-on:<br>Rogue Wave IMSL* Library | Add-on:<br>Rogue Wave IMSL* Library | Add-on:<br>Rogue Wave IMSL* Library |

Additional configurations including, floating and academic, are available at: http://intel.ly/perf-tools

# INTEL® MATH KERNEL LIBRARY (INTEL® MKL)

### Linear Algebra
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- PARDISO* SMP & Cluster
- Iterative sparse solvers

### Fast Fourier Transforms
- Multidimensional
- FFTW interfaces
- Cluster FFT

### Vector Math
- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

### Deep Neural Networks
- Convolution
- Pooling
- Normalization
- ReLU
- Inner Product

### Summary Statistics
- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

### And More
- Vector RNGs
- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

# WHAT'S NEW IN INTEL® MKL V.2019 BETA

- **Introduced new functions to JIT (create) optimized S/DGEMM-like matrix multiply kernels for small matrix sizes (m,n,k <=16)**
- Introduced Extreme{EVD/SVD} functionality
- Introduce SparseQR functionality
- Introduced Multinominal Random Number Generators
- Improved performance of 1D/3D  FFT

# INTEL® MKL – BLAS JIT API

- Types

```
typedef enum {MKL_JIT_ERROR, MKL_JIT_SUCCESS, MKL_NO_JIT} mkl_jit_status_t;

typedef (*{s,d}gemm_jit_kernel_t)(void*, FP_TYPE*, FP_TYPE*, FP_TYPE*)
```

- **Functions:**

```
mkl_jit_status_t mkl_jit_create_{s,d}gemm(void** jitter, MKL_LAYOUT
layout, MKL_TRANSPOSE transa, MKL_TRANSPOSE transb, MKL_INT m, MKL_INT n, MKL_INT k,
FP_TYPE alpha, MKL_INT lda, MKL_INT ldb, FP_TYPE beta, MKL_INT ldc)

{s,d}gemm_jit_kernel_t mkl_jit_get_{s,d}gemm_ptr(void* jitter)

mkl_jit_status_t mkl_jit_destroy(void* jitter)
```

# INTEL® MKL – BLAS JIT , DETAILS

- Language supported: C (CBLAS interface only) and Fortran (same function name as C API)
- All architectures supported, by default pointer to standard GEMM is returned
- JIT only for AVX2, AVX512 and M, N, K ≤ 16
- MKL_DIRECT_CALL_JIT
- Return status of mkl_jit_create_?gemm
  - Memory allocation fails: MKL_JIT_ERROR
  - JIT happened: MKL_JIT_SUCCESS
  - Standard GEMM is returned: MKL_NO_JIT
- Limitations:  CNR features are not supported

# INTEL® MKL – BLAS JIT , EXAMPLE

```
int main() {
    MKL_INT m = 10, n = 5, k = 12, lda = 32, ldb = 32, ldc = 32;
    MKL_TRANSPOSE transa = MKL_NOTRANS, transb = MKL_TRANS;
    MKL_LAYOUT layout = MKL_COL_MAJOR;
    float alpha = 2.0, beta = 1.0;
    float *a, *b, *c;
    void* jitter_s_10_5_12;

    // allocate and initialize matrices
    mkl_jit_status_t status = mkl_jit_create_sgemm(&jitter_s_10_5_12, layout, transa, transb, m, n, k,
                                                   alpha, lda, ldb, beta, ldc);
    if (MKL_JIT_ERROR == status) {
        printf("Creation jitter failed\n");
        return 1;
    }
    sgemm_jit_kernel_t sgemm_10_5_12 = mkl_jit_get_sgemm_ptr(jitter_s_10_5_12);

    sgemm_10_5_12(jitter_s_10_5_12, a, b, c);

    mkl_jit_destroy(jitter_s_10_5_12);
    // free matrices
    return 0;
}
```

perform C = alpha * A x B + beta*C

# INTEL® MKL – BLAS JIT USAGE, ACTIVITY

- Open and review C examples:
  - small_gemm.c  jit_small_gemm.c  makefile
- Set compiler's environment:
  - sourse opt/intel/compilers_and_libraries_2018/linux/bin/**compilervars.sh intel64**
- Compiling and Linking :

  icc –mkl small_gemm.c

  icc -DMKL_DIRECT_CALL -std=c99 -I${MKL_INCL}  -mkl  small_gemm.c
- Refer to MKL Linker Adviser :  https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor

# INTEL® MKL – BLAS JIT USAGE, ACTIVITY

#Building:

make

#Run and record the Executions times:

./run.sh

Note: be aware that CPU supports AVX2 and or AVX-512 ISA:

cat /proc/cpuinfo | grep avx2 ( avx512)

```
C:\AR\JIT_Demo\res.txt
[gfedorov@skl3 JIT_Demo]$ ./run.sh
[2 x 2], SGEMM Execution Time          == 1.769513e-07
[2 x 2], JIT_SGEMM Execution Time      == 4.656613e-08

[3 x 3], SGEMM Execution Time          == 1.750886e-07
[3 x 3], JIT_SGEMM Execution Time      == 4.656613e-08

[4 x 4], SGEMM Execution Time          == 1.806766e-07
[4 x 4], JIT_SGEMM Execution Time      == 4.097819e-08

[6 x 6], SGEMM Execution Time          == 1.825392e-07
[6 x 6], JIT_SGEMM Execution Time      == 5.029142e-08

[8 x 8], SGEMM Execution Time          == 1.918525e-07
[8 x 8], JIT_SGEMM Execution Time      == 4.656613e-08

[12 x 12], SGEMM Execution Time        == 2.142042e-07
[12 x 12], JIT_SGEMM Execution Time    == 7.078052e-08

[16 x 16], SGEMM Execution Time        == 2.272427e-07
[16 x 16], JIT_SGEMM Execution Time    == 8.195639e-08

[20 x 20], SGEMM Execution Time        == 3.259629e-07
Error:  mkl_no_jit, exit
```

# INTEL® MKL – BLAS JIT USAGE, ACTIVITY -- DIRECT_CALL

## MKL_DIRECT_CALL_JIT :

- make jitdirect

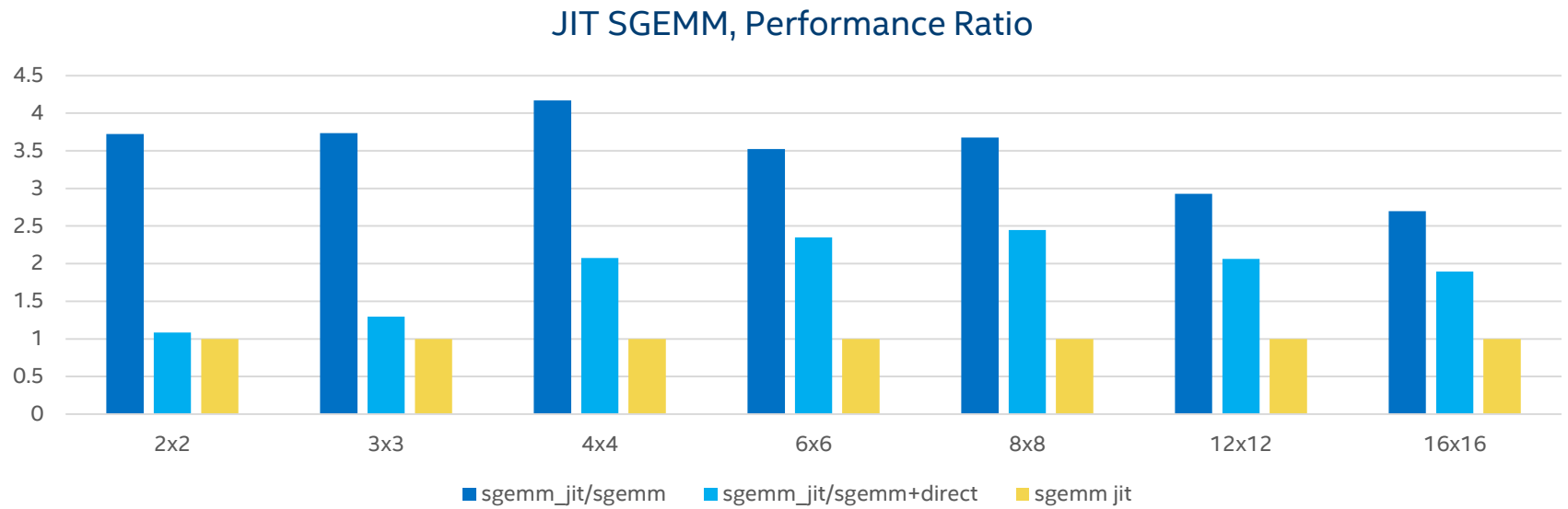
```
icc -DMKL_DIRECT_CALL_JIT -std=c99 -I${MKL_INCL} small_gemm.c -o jit_direct.out  -Wl,--
start-group  ${2019}/mkl/lib/intel64/libmkl_intel_lp64.a
${2019}/mkl/lib/intel64/libmkl_intel_thread.a ${2019}/mkl/lib/intel64/libmkl_core.a  -Wl,--
end-group -liomp5 -lpthread -lm -ldl
```

- run: ./jitdirect.out <size>,  size = {2, 3, 4, 6, 8, 12, 16, 20}
- Compare the performance results with previous calls

Conclusions? Do you see something like this?

```
[4 x 4], SGEMM       Execution Time  ==  17. 69513e-08
[4 x 4], SGEMM JIT   Execution Time  ==  7.078052e-08
[4 x 4], JIT_SGEMM Execution Time  ==  4.097819e-08
```

# INTEL® MKL – BLAS, PERFORMANCE, SMALL SIZES




Configuration Info – SW Versions: Intel® Math Kernel Library (Intel® MKL) 2019. Hardware: Intel(R) Xeon(R) Platinum 8168 CPU @ 2.70GHz ,192 GB RAM (12x16GB DDR4-2666). 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. Other brands and names are the property of their respective owners. Benchmark Source: Intel Corporation

**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 .

# Legal Disclaimer & Optimization Notice

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.

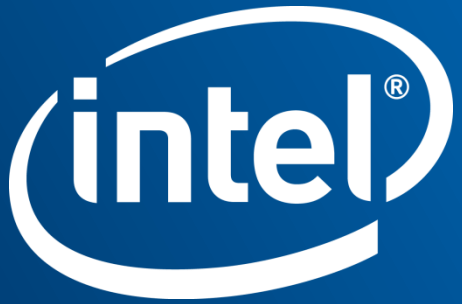
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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.



**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


intel®