INTEL® MKL - FAST FOURIER TRANSFORM (FFT)

Gennady Fedorov - Technical Consulting Engineer

Intel Architecture, Graphics and Software (IAGS)

PRACE workshop, June 2020

Gennady.Fedorov@intel.com

Intel® Math Kernel Library

Linear Algebra
- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Iterative sparse solvers
- PARDISO*
- Cluster Sparse Solver

FFT
CFFT

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

Removed since MKL v.2020

Vector RNGs
- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

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

Benchmarks
- Intel(R) Distribution for LINPACK* Benchmark
- High Performance Computing Linpack Benchmark
- High Performance Conjugate radient Benchmark

Intel® Architecture Platforms

Operating System: Windows*, Linux*, MacOS[1]*

[1] Available only in Intel® Parallel Studio Composer Edition.

Intel MKL FFT - Agenda

- **Introduction**
- **FFT API**
- **Demo – General Case, Usage Modes**
- **Demo -** 1d-2d case, Batch mode
- **Demo – MKL FFT -** FFTW

Introduction

- 1, 2 & 3 dimensional (up to the order of 7)
- Multithreaded
- Mixed radix
- Single and double precision complex and real transforms
- Placement of results: in-place, out of place
- Non-unit stride distribution of data within each data set
- User-specified scaling, transform sign
- Multiple one-dimensional transforms on single call
- Supports FFTW interface through wrappers

Introduction, Cluster FFT

- These functions are available only for Intel® 64
- Works with MPI using BLACS
- 1, 2, 3 and multidimensional (up to the order of 7)
- Require basic MPI programming skills
- Supported Intel® MPI, Open MPI, MPICH and SGI MPT
- Same interface as the DFT from standard MKL

MKL DFTI API

Overview

- DFTI_DESCRIPTOR_HANDLE
- 5 base functions: Create, Adjust(optional), Commit, Compute, Free
- numerous configuration parameters

See also :

http://portal.acm.org/citation.cfm?id=1114271)

DFTI_PRECISION

DFTI_DIMENSION, DFTI_LENGTHS

DFTI_PLACEMENT

DFTI_THREAD_LIMIT

DFTI_INPUT_STRIDES, FTI_OUTPUT_STRIDES

DFTI_NUMBER_OF_TRANSFORMS

DFTI_COMPLEX_STORAGE

DFTI_REAL_STORAGE

DFTI_CONJUGATE_EVEN_STORAGE

……..

MKL DFTI interface routines

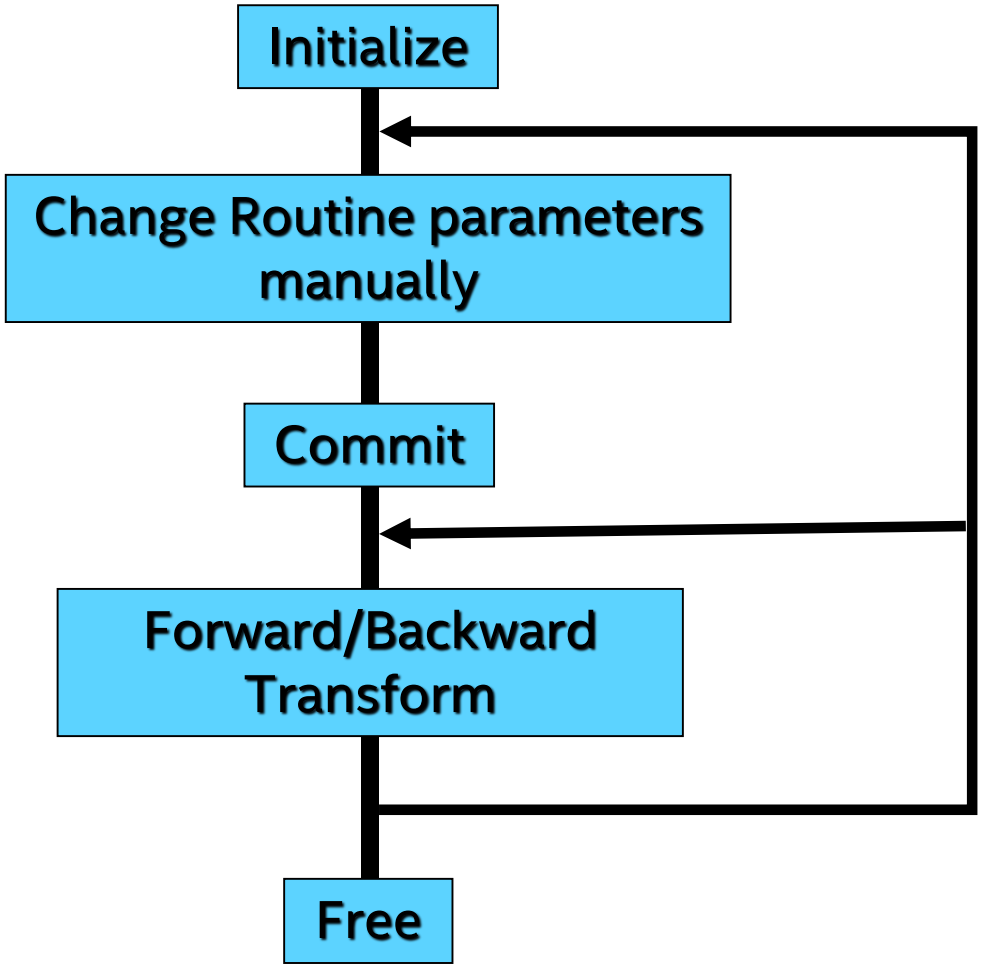
```
DftiCreateDescriptor

DftiSetValue

DftiCommitDescriptor

DftiComputeForward
DftiComputeBackward

DftiFreeDescriptor
```



MKL DFTI API, example

Complex-to-complex 1D transform for double precision data not inplace.

```
/* Create Dfti descriptor for 1D double precision  transform */
Status = DftiCreateDescriptor( &Desc_Handle, DFTI_DOUBLE, DFTI_COMPLEX, 1, n );

/* Set placement of result DFTI_NOT_INPLACE */
Status = DftiSetValue(Desc_Handle, DFTI_PLACEMENT, DFTI_NOT_INPLACE);

/* Commit Dfti descriptor */
Status = DftiCommitDescriptor( Desc_Handle );

/* Compute Forward transform */
Status = DftiComputeForward(Desc_Handle, x_in, x_out);

/*  Free DFTI descriptor */
Status = DftiFreeDescriptor(&Desc_Handle);
```

Requirements

- Intel® Parallel Studio XE 2020 Composer Edition with Intel® C++ Compiler
- Linux* OS supported by Intel® C++ Compiler
- Recommended to have at least 3nd **generation Intel® Core™ processor** (with Intel® AVX2)
- Setting the PATH, LIB, and INCLUDE environment variables

Compiler:

source /opt/intel/compilers_and_libraries_2020.1.127/linux/bin/compilervars.sh intel64

module load intel64/19.1up01 // ssh Meggie, PRACE Workshop

MKL:

or source <mklroot>/bin/mklvars.sh intel64

check the version: **echo $MKLROOT** → /opt/intel/compilers_and_libraries_2020.1.217/linux/mkl/

Demo – General Case, 1D FFT, in-place

Directory: ~/workshop/mkl/**FFT**

- Review test: **test_dft_1d.c**
- Compiling: **icc –mkl test_dft_1D.c**
- **export MKL_NUM_THREADS=1**
- **./a.out 10**
- Outputs**:**

```
[gfedorov@skx2 4FFT]$ ./a.out 10
DFTI_LENGTHS       = {10000000}
ExecTime        == 2.862838,sec
Performance   ==  4.47 GFlops...
```

* -- Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz ,192 GB RAM

Demo – General Case, 1D FFT, Usage Modes

Verbose Mode

- export MKL_VERBOSE=1
- $ ./a.out 10
- **Output**:

MKL_VERBOSE Intel(R) MKL 2020.0 Update 1 Product build 20200208 for Intel(R) 64 architecture Intel(R) Advanced Vector Extensions 512 (Intel(R) AVX-512) enabled processors, Lnx 2.40GHz intel_thread

MKL_VERBOSE FFT(**dcfi**10000000,tLim:1,desc:0x1fd3e40) 259.66ms CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:1
ExecTime == 2.847701,sec
Performance == 4.49 GFlops...

Demo – General Case, 1D FFT, Threading, OMP

OpenMP: mkl_intel_thread.[lib,dll], libmkl_intel_thread.[a,so]

TBB : mkl_tbb_thread.lib[lib,dll], libmkl_tbb_thread.[a,so]

Sequential: mkl_sequential.[lib,dll], libmkl_sequential.[a.so]

Sequential mode: ./a.out {1, 10, 100} (export MKL_NUM_THREADS=1)

./a.out 1 …Performance = 6.2 GFlops…
./a.out 10 …Performance = 2.9 GFlops…
./a.out 100 …Performance = 4.8 GFlops…

Scaling: Review and **./run_ompthr_scaling.sh** (icc –mkl=parallel test.c or unset MKL_NUM_THREADS)

Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz ,192 GB RAM results:

size*10^6/threads	1	2	4	8	16	32
100	4.8	5.8	11.4	20.7	35.1	43.6
10	4.5	6.0	11.6	21.1	31.4	40.0
1	6.4	7.8	14.6	22.5	40.9	58.3

Demo – General Case, 1D FFT, Threading, TBB

cd ../tbb (Directory: ~workshop/mkl/FFT/tbb)

- review makefile:

```
-Wl,--start-group \
${MKLROOT}/lib/intel64/libmkl_intel_lp64.a \
${MKLROOT}/lib/intel64/libmkl_tbb_thread.a \
${MKLROOT}/lib/intel64/libmkl_core.a \
-Wl,--end-group -ltbb -lstdc++ -lpthread -lm -ldl
```

- make

- ./run.sh

- Observation ☺?

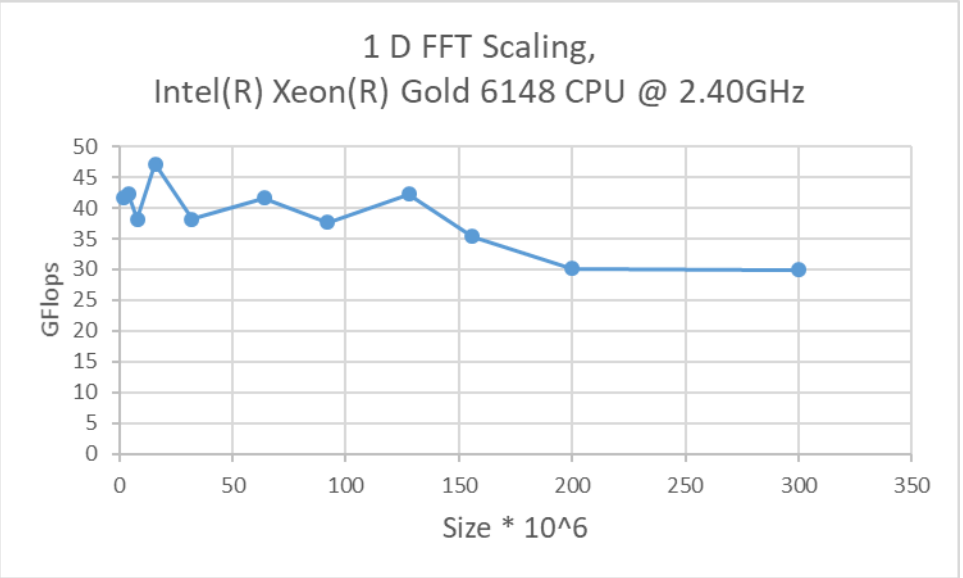
Demo – General Case, 1D FFT, Size Scaling

Problem Size Scaling

cd .. (Directory: <mkl_workshop>/FFT)

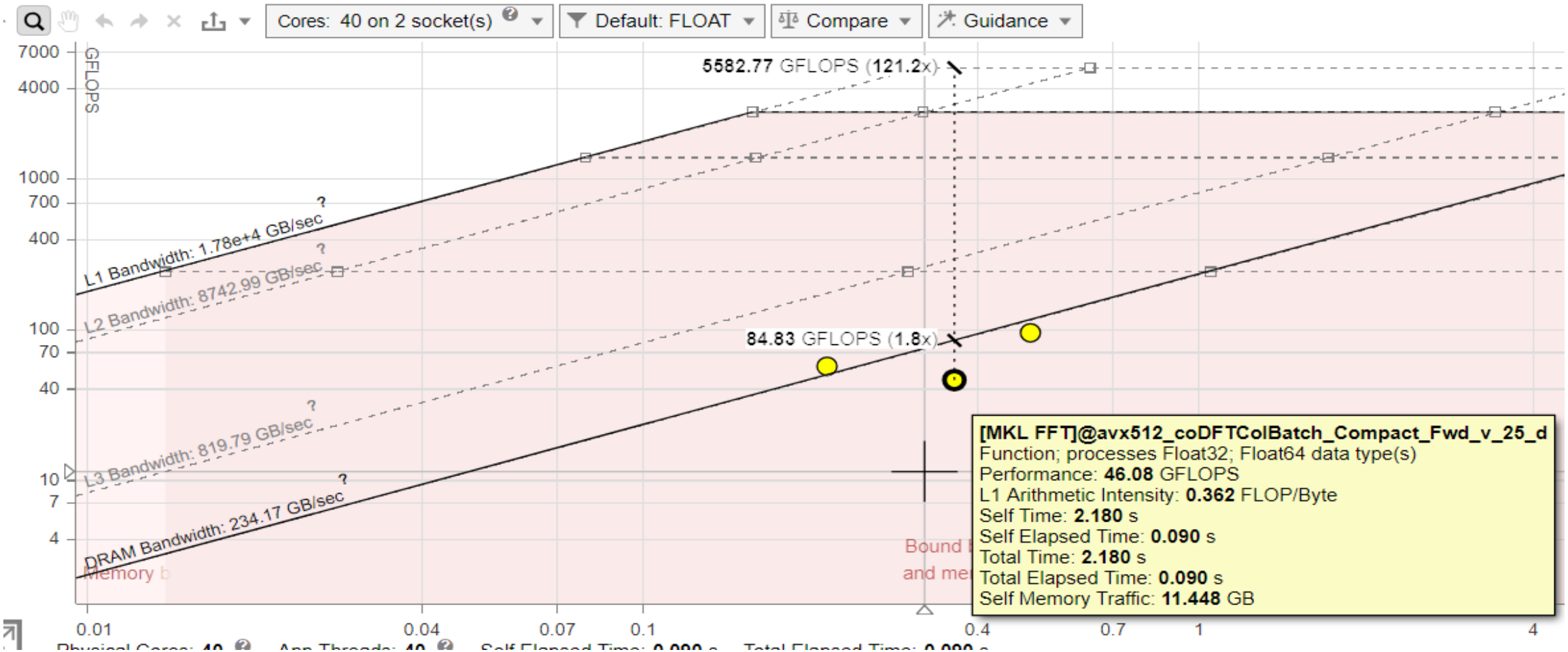
- Review run_problem_size_scaling.sh
- icc –mkl test_dft_1d.c
- ./run_problem_size_scaling.sh

Do you see smth like as follows:



Solution - Cluster FFT!

Intel® Adviser, Roofline Analysis. Memory bound



Demo – 2DFFT

2D FFT using batch of 1D FFT or single 2DFFT calls

➢ **Review test_fft_2d_by_1d.c**

```
DftiCreateDescriptor(&hand_x, DFTI_DOUBLE, DFTI_COMPLEX, 1, M);
DftiSetValue(hand_x, DFTI_NUMBER_OF_TRANSFORMS, N )
DftiSetValue(hand_x, DFTI_INPUT_DISTANCE, M);

DftiCreateDescriptor(&hand_y, DFTI_DOUBLE, DFTI_COMPLEX, 1, M);
DftiSetValue(hand_y, DFTI_NUMBER_OF_TRANSFORMS, N )
DftiSetValue(&hand_y, DFTI_INPUT_DISTANCE, M);

DftiComputeForward(hand_x, data);
DftiComputeForward(hand_y, data);

DftiFreeDescriptor(&hand_x);
DftiFreeDescriptor(&hand_y);
```

```
Int clengths[2] = {M, N};
DftiCreateDescriptor
(&hand_xy, DFTI_DOUBLE, DFTI_COMPLEX, 2, clengths);

DftiSetValue
(hand_xy, DFTI_NUMBER_OF_TRANSFORMS, 1);

DftiComputeForward (hand_xy, data);

DftiFreeDescriptor(&hand_xy);
```

Demo – 2DFFT, cont.

- **icc -mkl test_fft_2d_by_1d.c**
- **./a.out**

MKL_VERBOSE Intel(R) **MKL 2020.0** Update 1 Product build 20200208 for Intel(R) 64 architecture Intel(R) Advanced Vector Extensions 512 (Intel(R) **AVX-512**) enabled processors, Lnx 2.40GHz intel_thread
MKL_VERBOSE FFT(dcfi1920x1200,tLim:40,desc:0x101fe40) 39.31ms CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:40
MKL_VERBOSE FFT(dcfi1200x1920,tLim:40,desc:0x103c180) 2.96ms CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:40
Verify the result, errthr = 2.35e-14
Verified, maximum error was 3.29e-16
MKL_VERBOSE FFT(dcfi1200x1920,tLim:40,desc:0x1043400) 2.34ms CNR:OFF Dyn:1 FastMM:1 TID:0 NThr:40
Verify the result, errthr = 2.35e-14
Verified, maximum error was 3.29e-16
Execution time of **1D calls** == **4.243274e-02**
Execution time of **2D call** == **2.373712e-03**

FFTW API Support

Intel MKL supports FFTW2 and FFTW3 APIs

FFTW3 API:

- interfaces are integrated in Intel MKL by default

- Option to build – see interfaces/fftw*/makefiles

FFTW2 – are not integrated: Build standalone library of FFTW2 C/F wrappers to Intel(R) MKL.

MKLROOT/interfaces: fftw2xc, fftw2xf, fftw2x_cdft, fftw3xc, fftw3xf, fftw3x_cdft

Examples: MKLROOT\examples:

fftw2xc, fftw2xf, fftw2x_cdft, fftw3xc, fftw3xf, fftw3xf_cdft and fftw3x_cdft

Note: The FFTW2 and FFTW3 interfaces are not compatible with each other. Avoid linking to both of them.

DEMO – FFTW, FFT

Directory: ~/workshop/mkl/FFT/**fftw**

- prebuilt - fftw-3.3.7

➢ Review **fftw.c**, **mkl_fftw.c** and **makefile**

  ➢ make

  ➢ export LD_LIBRARY_PATH=.:$LD_LIBRARY_PATH;

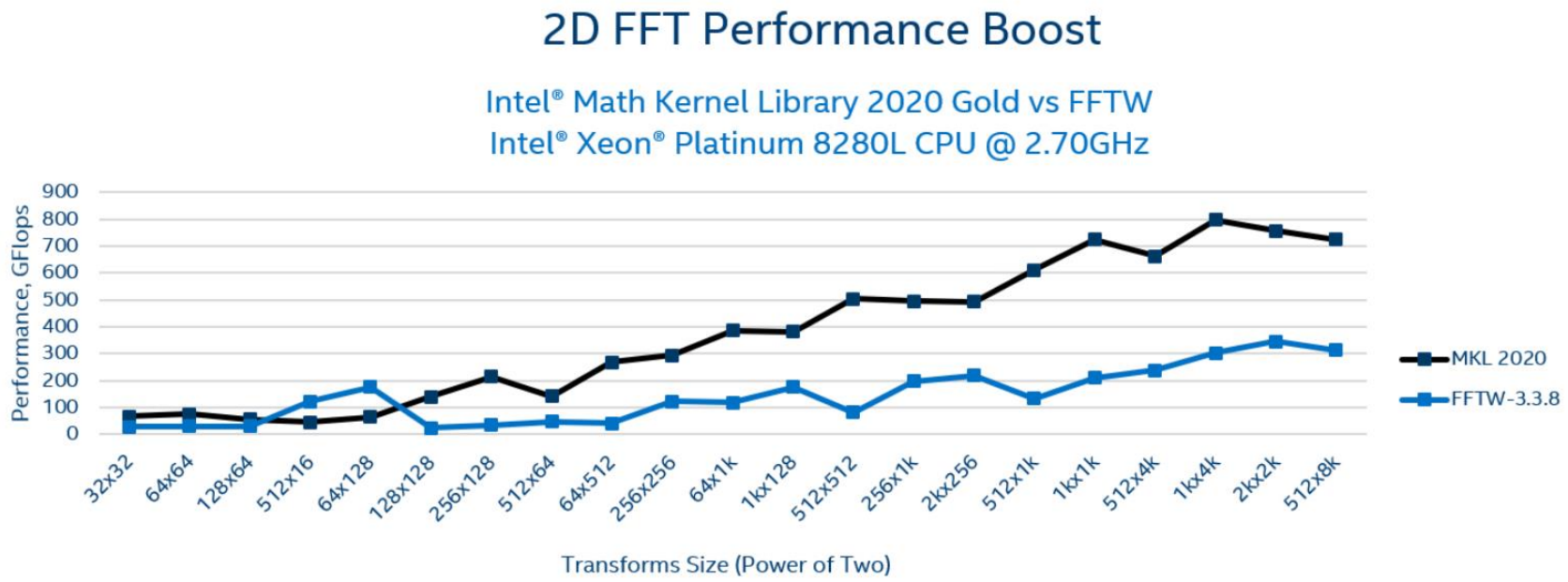
  ➢ ./run.sh

➢ **Observation ?**

➢ export MKL_VERBOSE and ./run.sh

MKL 2020 – FFTW, FFT, benchmarking

https://software.intel.com/en-us/mkl/features/benchmarks



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.
Performance results are based on testing as of **November 5, 2019** and may not reflect all publicly available security updates. See configuration disclosure for details. No product can be absolutely secure.

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.

Configuration: Testing by Intel as of **November 5, 2019**. Intel® Xeon® Platinum 8280L 2x28@2.7GHz 192GB DDR4-2666 using Intel® Math Kernel Library 2020.

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

Performance Tips

- KMP_AFFINITY=compact, granularity=fine
- MKL_DYNAMIC=false
- MKL_NUM_THREADS *varies*
- Align data
  - help vector load/store
  - Avoid cache-thrashing alignments (e.g. 2Kx2K)
- Use batched transformation where possible
- Know optimize radices: 2, 3, 5, 7, 11, 13

Intel MKL Resources

Intel® MKL website:

– https://software.intel.com/en-us/intel-mkl

Intel MKL forum:

– https://software.intel.com/en-us/forums/intel-math-kernel-library

Intel® MKL benchmarks:

– https://software.intel.com/en-us/intel-mkl/benchmarks#

Intel® MKL link line advisor:

– http://software.intel.com/en-us/articles/intel-mkl-link-line-advisor/

Legal Disclaimer & Optimization Notice

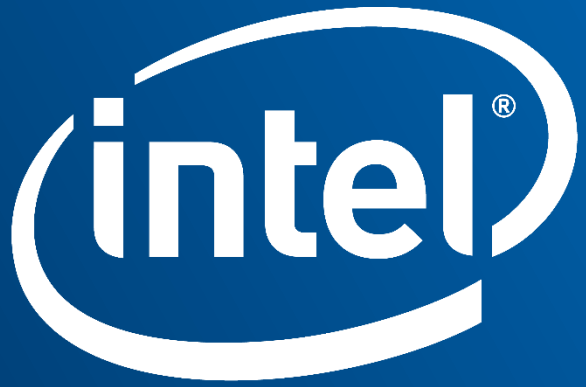
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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



Software