WHITE PAPER

Engineering and Modeling
Intel® Math Kernel Library

Run Your Ansys® Fluent® Simulations at Top Speed

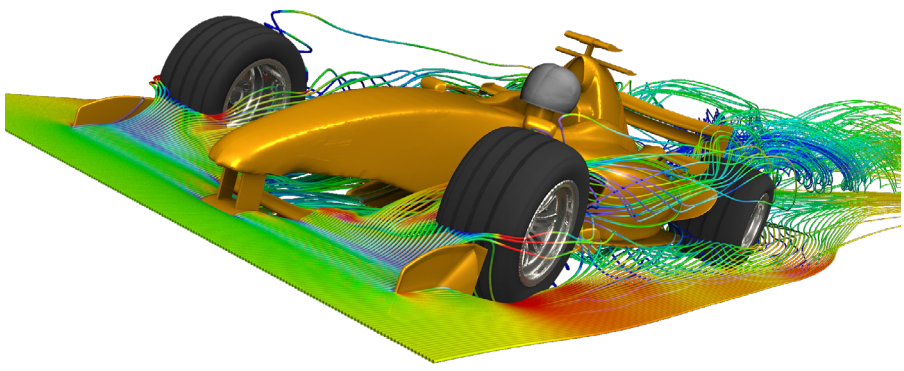
Ansys Fluent takes advantage of Intel Advanced Vector Extensions (Intel AVX) and Intel Math Kernel Library (Intel MKL) to enable greater efficiencies and optimized performance.

Addressing the Need for Speed with Intel® Xeon® Scalable Processors

Ansys Fluent is a versatile computational fluid dynamics (CFD) software tool widely used by academia and industry in high-performance computing (HPC) environments. The tool includes validated physical modeling capabilities that deliver accurate results across a wide range of CFD and multiphysics applications. The Ansys Fluent solver can be so computationally demanding that complex simulations can take hours—or sometimes even days—to complete. Therefore, Ansys Fluent users are always eager for performance improvements that will help reduce those long wait times.

Intel Xeon Scalable processors offer unique capabilities—including instruction sets and libraries—that can be used to optimize compute-intensive operations in Ansys Fluent and thereby improve product performance. Ansys and Intel work together to deliver the best possible performance to Ansys Fluent users on Intel architecture. One example of this ongoing collaboration is the development of a new solver, currently in preview, that makes use of Intel Advanced Vector Extensions 512 (Intel AVX-512) instruction sets and Intel Math Kernel Library (Intel MKL) to optimize performance on Intel-based multicore clusters.

This paper reviews performance testing results in Ansys Fluent 2020 R1 when making use of the Intel MKL sparse LDU smoother, and it explains how the improved performance results are accomplished. If you are interested in trying the solution yourself, follow the instructions in Appendix 1 to enable this preview feature in your own practical implementations. You can also find a deep dive into the inner workings and interface of the Intel sparse LDU smoother in Appendix 2.



Key Terms

Solver: Software code that solves a problem or problems through mathematical calculations.

Sparse solver: A linear solver that has been optimized for sparsely populated matrices, like those found in finite element analysis (FEA).

Algebraic multigrid (AMG) solver: A type of solver that solves differential equations using a hierarchy of discrete functions, models, variables and equations.

Smoother: An algorithm that provides an approximate solution to a specific piece of the overall set of equations and allows important patterns to stand out. Smoothers are an important part of an AMG solver.

Incomplete lower/upper (ILU)/LDU smoother: The ILU smoother is one of the native Ansys Fluent smoothers. The LDU smoother is the Intel MKL smoother that replaces the ILU smoother in this testing to improve performance.

Intel Advanced Vector Extensions (Intel AVX-512 and Intel AVX2): A set of instructions for performing single instruction, multiple data (SIMD) operations on Intel processors, improving performance for large datasets. Intel AVX2 includes 256-bit integer instructions and supports floating-point fused multiply-add instructions. Intel AVX-512 enables twice the number of floating-point operations per second per clock cycle compared to Intel AVX2 by decreasing latency.

Intel Math Kernel Library (Intel MKL): A math computing library of optimized, threaded routines for linear algebra, fast Fourier transforms (FFTs), vector math, random number generators and direct and iterative sparse solvers.

Performance Results

Intel conducted two kinds of testing to evaluate performance improvements to Ansys Fluent using the Intel MKL sparse LDU smoother: the Ansys Fluent benchmark suite and Ansys Fluent built-in software profiles.

Ansys Fluent Software Benchmark Suite

Intel measured performance improvements on a subset from the Ansys Fluent benchmark suite using a version of the Intel MKL sparse LDU smoother interface available in Ansys Fluent 2020 R1. Tables 1 and 2 show relative solver ratings in single-node and eight-node hardware configurations respectively, each comparing four software variations:

- Fluent baseline binaries with the Fluent smoother
- Fluent baseline binaries with the Intel MKL sparse LDU smoother
- Fluent optimized binaries (-platform=Intel) with the Fluent smoother
- Fluent optimized binaries with Intel the MKL sparse LDU smoother

Table 1. Ansys Fluent 2020 R1 performance with the Intel MKL sparse LDU smoother (single-node) running the Ansys Fluent benchmark suite[1]

Single-Node: Intel Xeon Platinum 8280L Processor		Ansys Fluent Relative Solver Rating Normalized to Fluent Baseline Binaries with a Native Smoother			
Case	N Core Count	Fluent Baseline	Fluent Baseline Plus Intel MKL Sparse LDU Smoother	Fluent Optimized Binaries*	Fluent Optimized Binaries* Plus Intel MKL Sparse LDU Smoother
sedan_4m	56	1.00	1.04	1.01	1.04
aircraft_wing_14m	56	1.00	1.02	1.00	1.01
combustor_12m	56	1.00	1.05	1.08	1.13
pump_2m	56	1.00	1.11	1.02	1.11
rotor_3m	56	1.00	1.01	1.01	1.01
aircraft_wing_2m	56	1.00	1.00	1.00	0.99
exhaust_system_33m	56	1.00	1.05	1.00	1.05
landing_gear_15m	56	1.00	1.03	1.01	1.03

*Optimized binaries are selected with the optional -platform=intel Fluent command-line argument.

Table 2. Ansys Fluent 2020 R1 performance with the Intel MKL sparse LDU smoother (eight-node cluster) running the Ansys Fluent benchmark suite[1]

Eight-Node Cluster: Intel Xeon Platinum 8280L Processor		Ansys Fluent Relative Solver Rating Normalized to Fluent Baseline Binaries with a Native Smoother			
Case	N Core Count	Fluent Baseline	Fluent Baseline Plus Intel MKL Sparse LDU Smoother	Fluent Optimized Binaries*	Fluent Optimized Binaries* Plus Intel MKL Sparse LDU Smoother
sedan_4m	448	1.00	0.97	1.03	1.00
aircraft_wing_14m	448	1.00	1.01	1.00	0.98
combustor_12m	448	1.00	1.04	1.12	1.16
pump_2m	448	1.00	1.00	1.03	1.01
rotor_3m	448	1.00	0.98	1.03	0.99
aircraft_wing_2m	448	1.00	0.96	1.01	0.93
exhaust_system_33m	448	1.00	1.05	1.01	1.05
landing_gear_15m	448	1.00	1.02	1.02	1.02
combustor_71m	448	1.00	1.13	1.00	1.13
f1_racecar_140m	448	1.00	1.15	1.01	1.15
open_racecar_280m	448	1.00	1.08	1.01	1.09

*Optimized binaries are selected with the optional -platform=intel Fluent command-line argument.

The level of performance improvement found in this testing varies by test case, but a wide variety of test cases, including the most common solver options in Fluent, saw improvements of up to 13 percent in the single-node tests and 16 percent in the eight-node cluster. These results show that, in general, where using either the Fluent optimized binaries or the Intel MKL sparse LDU smoother is beneficial, the benefits of using both are complementary.[2]

The performance benefits of both the optimized "-platform=intel" option and the Intel MKL sparse LDU smoother can effectively scale to thousands of cores in large-scale runs. Profiling was recorded for the test case F1_racecar_140m on eight nodes of Intel Xeon Platinum 8280L processors, using 56 cores per node for a total of 448 cores.[3] These are top-down profiles where each function shown is a subcomponent of some or all of the functions above it.

Figures 1 and 2 illustrate the performance boost at increasing core counts on an Intel Xeon Platinum 8260L processor–based cluster with Intel Omni-Path Architecture (Intel OPA) interconnect. As shown in Figure 1, the test case F1_racecar_140m scales well to 6,144 cores on the cluster, with each node fully subscribed, and with Intel OPA interconnect and Intel MPI Library.[4]

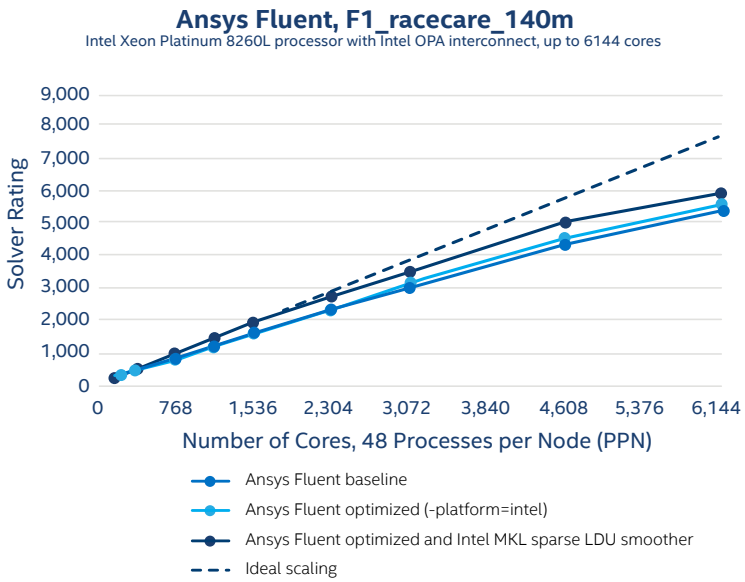


Figure 1. Racecar test case scaling up to 6,144 cores[4]

Figure 2 shows that the combined impact of the Intel MKL sparse LDU smoother and the "-platform=intel" optimized code path is a 12 percent to 19 percent improvement over the range shown.[4]

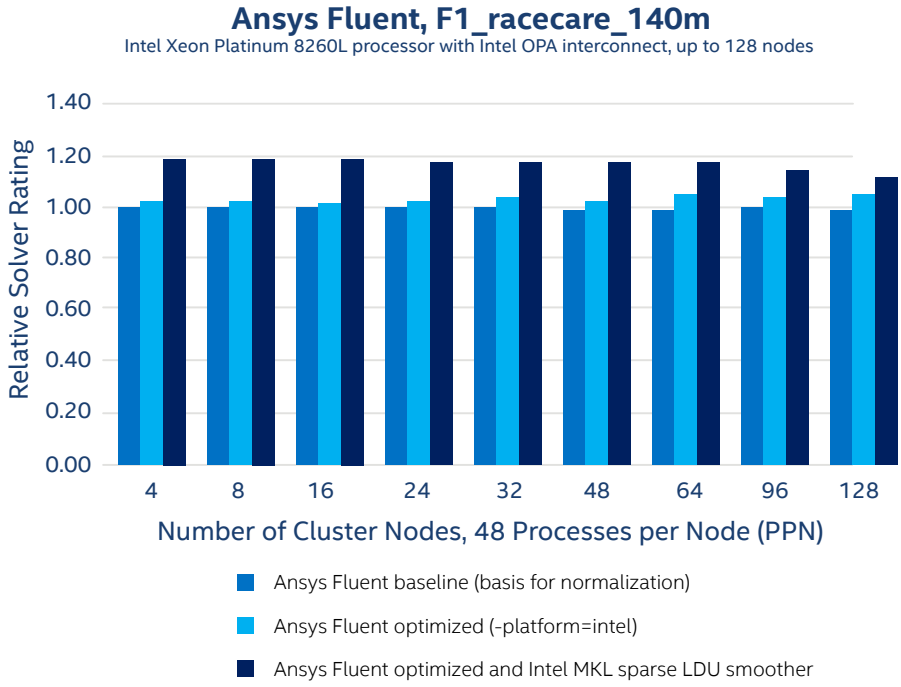


Figure 2. Racecar test case performance showed improvements of 12 to 19 percent[4]

Ansys Fluent Software Profiles

It is easy to generate software profiles from a Fluent execution by adding software profiling to the Fluent input command file. Table 3 shows profiles recorded for test case F1_racecar_140m on 8 nodes of Intel Xeon Platinum 8280L processors, using 56 cores per node for a total of 448 cores. These are top-down profiles, where each function shown is a subcomponent of some or all of functions above it.

Table 3. Profile before and after the introduction of the Intel MKL sparse LDU smoother

		Without Intel MKL Sparse LDU Smoother			With Intel MKL Sparse LDU Smoother		
Index	Function	N_0 Time	Minimum Rank Time	Maximum Rank Time	N_0 Time	Minimum Rank Time	Maximum Rank Time
0	literate	193.23	193.23	193.23	169.07	169.07	169.07
1	flow_iterate	193.22	193.22	193.22	169.07	169.07	169.07
2	solver_update	193.22	192.93	193.22	169.06	168.82	169.07
3	Flow	177.55	177.28	178.61	153.61	153.49	154.82
4	solve_pp	154.46	154.34	154.61	130.83	130.70	130.98
5	AMG_Solve	145.35	145.34	145.35	121.80	121.79	121.81
6	Smooth_ILU4	117.47	114.69	118.80			
6	Smooth_ILU_ud				93.36	90.87	95.56

Before the introduction of the Intel MKL sparse LDU smoother, the native Ansys smoother, Smooth_ILU4, was present in the profile, consuming 118 seconds out of 193 seconds of solver time. After its introduction, the Intel MKL sparse LDU smoother showed up in the software profile as Smooth_ILU_ud, and time consumption was reduced to 96 seconds, an improvement of 19 percent for that function.

Understanding the Test Results

The Ansys Fluent solver uses an iterative method known as algebraic multigrid (AMG) for a sparse system of equations. The AMG solver is highly efficient in terms of solution speed, robustness and memory footprint, and it is no simple matter to further accelerate with Intel AVX instruction sets.

A key component of the Ansys Fluent AMG solver is the block-method incomplete lower/upper (ILU) decomposition smoother for the transitions between multigrid levels. The ILU smoother itself experiences minimal benefits from vectorization, owing to indirect memory access, high memory bandwidth demand and loop-carried dependencies. Compiler optimizations using Intel AVX2 instructions can provide users with a small boost in performance—a few percent on average and up to 8 percent or more in some cases.[1] This small but significant performance enhancement is exploited in the Fluent optimized binaries, as selected with the " platform=intel" option.

With the release of Ansys Fluent 2020 R1, Ansys has made available a preview version of a solver pathway that takes advantage of Intel MKL Inspector-Executor Sparse BLAS (Intel MKL IE SpBLAS) functionality for the ILU smoother. This functionality, referred to as the Intel MKL sparse LDU smoother, provides an additional acceleration of up to 15 percent, and it is complementary to the "-platform=intel" code path in cases where both are beneficial.[1] (See Appendix 2 for more on the Intel MKL IE SpBLAS functionality introduced in the 2020 release of Intel MKL.)

Working Together to Fine-Tune Performance

Ansys Fluent using the Intel MKL sparse LDU smoother and Fluent optimized "-platform=intel" code path demonstrates a performance boost of 12 to 19 percent.[4] This set of optimizations can make a big difference to designers and engineers running multiple simulations in a day. Together, Ansys and Intel continue to improve HPC applications that take advantage of ever-increasing core counts for faster simulations and a shorter time to market.

Learn More and Try the Preview

Learn more about Ansys and Intel

Learn more about **Intel MKL** and **Intel AVX-512**

Try out the Intel MKL sparse LDU smoother preview and see it in action (see **Appendix 1**).

Appendix 1: How to Try the Intel MKL Sparse LDU Smoother Yourself

This appendix demonstrates how to choose the Intel MKL sparse LDU smoother in an Ansys Fluent execution, both for single-precision and double-precision executions. Note that this process will be simplified considerably in future releases.

The interface for the Intel MKL sparse LDU smoother is implemented in two separate libraries, which are included with the Fluent 2020 R1 release. These libraries support either single-precision or double-precision execution. Depending whether Fluent is going to run with single or double precision, the user must identify which library is needed:

- Single precision: *<ANSYS_HOME>*/v201/fluent/lib/lnamd64/libmklsmoother_sp.so
- Double precision: *<ANSYS_HOME>*/v201/fluent/lib/lnamd64/libmklsmoother_dp.so

Each interface library contains the interface routine "ilu" as a member. Modify the Fluent installation as follows. (Note that, for this step, it might be best to make a separate copy of the installation for modification.)

1. Under the Fluent installation, locate Intel MKL libraries that were bundled with Ansys products.

   ```
   <ANSYS_HOME>/v201/tp/IntelMKL
   ```

2. Create a new subdirectory under **IntelMKL** to store the Intel MKL 2020 libraries, and name the subdirectory **2020.0.166**:

   ```
   cd <ANSYS_HOME>/v201/tp/IntelMKL

   mkdir 2020.0.166

   mkdir -parents 2020.0.166/linx64/lib/intel64/
   ```

3. Copy the Intel MKL 2020 gold release libraries into the Fluent installation directory:

   ```
   cp -p <MKL_HOME>/mkl/lib/intel64/*.so 2020.0.166/linx64/lib/intel64/
   ```

4. Locate the Fluent executable wrapper script to make a change to the version of Intel MKL referenced:

   ```
   <ANSYS_HOME>/v201/fluent/fluent20.1.0/bin/fluent
   ```

5. Modify the following line in that script from:

   ```
   sys_prepend_ld_library_path
   $FLUENT_INC/../tp/IntelMKL/2017.6.256/linx64/lib/intel64
   ```

   to:

   ```
   sys_prepend_ld_library_path
   $FLUENT_INC/../tp/IntelMKL/2020.0.166/linx64/lib/intel64
   ```

6. Finally, in the working directory for the simulation, modify the input command file, or .jou file, to call the ilu interface routine as a user-defined smoother. Add the following line to the input before the solve step:

   ```
   (rpsetvar 'amg-coupled/user-defined-smoother ilu@libmklsmoother_sp.so )
   ```

   Note that the line above references the single precision (_sp.so) version. Use double precision (_dp.so) if appropriate. Also, the line assumes that a copy of libmklsmoother_sp.so exists in the local working directory. You can add a full path to reference the Fluent installation instead with the following:

   ```
   (rpsetvar 'amg-coupled/user-defined-smoother ilu@ANSYS_HOME/v201/fluent/lib/lnamd64/
   libmklsmoother _sp.so )
   ```

Appendix 2: Deeper Dive into the Intel MKL Sparse LDU Smoother and Intel MKL Inspector-Executor Sparse BLAS (Intel MKL IE SpBLAS)

This appendix explains in greater detail the workings of the Intel MKL sparse LDU smoother. Detailed documentation for Intel MKL APIs is available at https://software.intel.com/en-us/mkl-developer-reference-c.

Intel MKL IE SpBLAS Overview and Details on the Sparse LDU Smoother

Intel MKL IE SpBLAS offers a rich set of sparse linear algebra functions, including some sparse solvers, and it employs an inspector-executor paradigm that separates operations (APIs) into two stages: analysis and execution. For a given matrix, the analysis would typically be called up to one time, whereas the execution might be called multiple times. During the analysis stage, the API inspects the matrix properties, including size, sparsity pattern and available parallelism, and it can apply matrix format/structure changes to the target or enable a more optimized algorithm. In the execution stage, multiple routine calls can take advantage of the analysis-stage data in order to improve performance. The Intel MKL IE SpBLAS model, with an optimize step separate from the execute step, allows Intel MKL to make use of Intel AVX-512 instruction sets in ways that were not possible previously, with only a single-stage execution model.

A common use case of this inspector-executor model is in an iterative solver where the matrix is not changing but the application of the matrix operation happens many times—typically until some sort of convergence happens. The inspector step's time cost is effectively amortized across the many execution calls, which can be sped up significantly by the choices made during inspection.

Starting with Intel MKL 2019 update 5 release, the library includes the sparse LDU smoother interface and the mkl_sparse_?_lu_smoother routine, which is supported for both single and double precision using real data types. However, this interface was not initially fully optimized for single precision. Intel MKL 2020 makes available the Intel MKL IE SpBLAS mkl_sparse_?_lu_smoother function that is now optimized for single and double precision, and Ansys has added an interface to the Ansys Fluent 2020 R1 solver to allow users to link against Intel MKL 2020 to try out the sparse LDU smoother function.

The Intel MKL sparse LDU smoother also follows the inspector-executor approach. The mkl_sparse_?_lu_smoother routine performs an update for an iterative solution x of the system Ax=b. It applies one iteration of an approximate preconditioner, which is based on the following representation: A ~ (L + D)*E*(U + D), where E is an approximate inverse of the diagonal (using the exact inverse will result in a symmetric Gauss-Seidel preconditioner), L and U are lower/upper triangular parts of A and D is the diagonal (the block diagonal in the case of a block compressed sparse row [BSR] format) of A (A = L + D + U). Essentially, this routine performs the following three operations:

- *r = b - A*x*
- *(L + D)*E*(U + D)*dx = r*
- *y = x + dx*

The pseudo-code below demonstrates the inspector-executor model and, in particular, the mkl_sparse_?_lu_smoother interface, in addition to the main ideas that were used for integration with the Ansys Fluent ILU smoother.

In order to work with Intel MKL IE SpBLAS, first you must create a handle for a matrix that stores the data internally and can be passed into the interface. Assuming that matrix A is stored in BSR format with a number of rows equal to nRows, a number of columns equal to nCols and square blocks of size block_size>=1, then bsrRowPtr, bsrColInd and bsrVal are arrays of row pointers, column indices and values respectively.

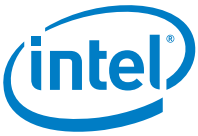
```
sparse_matrix_t bsrA;

mkl_sparse_s_create_bsr(&bsrA, SPARSE_INDEX_BASE_ZERO, SPARSE_LAYOUT_ROW_MAJOR, nRows,
nCols, block_size, bsrRowPtr, bsrRowPtr+1, bsrColInd, bsrVal);
```

You can then call the inspector stage (that is, add hints and call mkl_sparse_optimize). The main point of this stage is to provide a hint regarding the number of expected calls, the type of operation and the matrix structure. Then, you can perform the analysis and do appropriate optimizations, including, but not limited to, matrix conversion and thread-balancing. The handle will then store any optimizations made during the inspector step. The number of hints that can be set here is not limited, and its effect is cumulative. The inspector stage is optional and should be called only once.

```
sparse_operations_t op = SPARSE_OPERATION_NON_TRANSPOSE;

struct matrix_descr descr_gen;

descr_gen.type = SPARSE_MATRIX_TYPE_GENERAL;

mkl_sparse_set_lu_smoother_hint(bsrA, op, descr_gen, expected_calls);

mkl_sparse_optimize(bsrA);
```

The last step is to apply the smoother. This stage can be repeated as many times as necessary with the same matrix handle.

```
mkl_sparse_?_lu_smoother (op, bsrA, descr_gen, D, E, x, b)
```



[1] Based on Intel testing as of November 19, 2019. **Configurations:** *Single-node*: Intel Xeon Platinum 8280L processor running the Ansys Fluent benchmark suite with and without "-platform=intel" option and the Intel MKL sparse LDU smoother. *Eight-node cluster:* Intel Xeon Platinum 8280L processor running the Ansys Fluent benchmark suite with and without "-platform=intel" option and the Intel MKL sparse LDU smoother.

[2] The Intel AVX2–enabled code path option has been available with recent versions of Fluent and is enabled with the Fluent command-line option **-platform=intel**. This option speeds up execution time particularly for cases using the polyhedral cell type, like combustor_12m. The Intel MKL sparse LDU smoother option benefits cases that use Smooth_ILU4 calls within the Fluent solver. Those cases can be identified by adding software profiling to the Fluent journal input file, and then scanning the output to see if Smooth_ILU4 appears in the performance profile generated during a run.

[3] If you try the preview, you can test the performance this way on your own real-world models. These profiling results are easily generated from an Ansys Fluent execution.

[4] Based on Intel testing as of November 26, 2019. **Configurations:** *Endeavor cluster:* Intel Xeon Platinum 8260L processor (48 cores per cluster node) running the Ansys Fluent benchmark suite with and without "-platform=intel" option and the Intel MKL sparse LDU smoother, and with Intel OPA interconnect.

Printed in USA 0620/KD/PRW/PDF Please Recycle 342850-001US