

Tuning CFD applications for Intel[®] Xeon Phi[™] with TAU Commander and ParaTools ThreadSpotter

Izaak B. Beekman*, Nicholas Chaimov*, Sameer Shende*, Allen D. Malony*,
Nicholas Bisek[†], Ryan Gosse[†], and Andrew Wissink[‡]

*ParaTools, Inc.

Eugene, OR 97405 U.S.A.

{ibeekman, nchaimov, sameer, malony}@paratools.com

[†]Air Force Research Laboratory

Dayton, OH 45433, U.S.A.

{nicholas.bisek.1, ryan.gosse.1}@us.af.mil

[‡]CREATE-AV[™], U.S. Army

Mountain View, CA 94035, U.S.A.

andrew.m.wissink.civ@mail.mil

Abstract—The performance of two computation fluid dynamics (CFD) applications, FDL3DI and AFSI, running on the latest generation of Intel[®] Xeon Phi[™] processors is examined. FDL3DI is a finite difference, implicit large eddy simulation (ILES) program used at the Department of Defense. It is used to investigate a wide range of physical phenomena including the full HIFiRE 6 scramjet geometry. AFSI is an ablation fluid structure interaction CFD program being developed to simulate flow over ablative thermal protection systems in Earth's atmosphere. Two performance engineering tools are utilized in this work: TAU Commander, and ParaTools ThreadSpotter. TAU Commander implements a streamlined and partially automated performance engineering workflow and can highlight source regions limiting scalability through profiling and aggregate summary statistics. ParaTools ThreadSpotter analyzes an application as it is running and shows performance problems in order of importance. It provides a report with actionable recommendations correlated with source code lines enabling developers to fix the discovered problems. For both CFD applications, loop structure and poor memory access patterns appear to be the chief impediments to efficient execution on Xeon Phi[™] architectures.

I. EXTENDED ABSTRACT

The computational fluid dynamics (CFD) codes FDL3DI and AFSI are developed at the Air Force Research Laboratory. FDL3DI is a finite-difference implicit large eddy simulation (ILES) application that is applied to a wide range of challenging fluid dynamics simulations where detailed insight into flow physics is sought and Reynolds-averaged Navier-Stokes turbulence modeling is insufficient [1]–[6]. FDL3DI has been used for studying a wide variety of flow phenomena from dynamic stall on airfoils [7] to highly non-linear and complex shock boundary layer interactions with corner effects [2] and has even been used to simulate the entire HIFiRE 6 [8], [9] scramjet-like geometry. AFSI is a coupled ablation solver being developed at the Air Force Research Laboratory. AFSI stands for ablation fluid-structure interaction and is being developed to better predict the behavior of ablative thermal protection systems flying in the Earth's atmosphere. It includes

capabilities to use extant material response models developed by NASA, and is developing more detailed models tightly coupled with CFD [10]. AFSI uses multi-block structured meshes with parametric domain representation, allowing for the compact representation of mesh data, reducing the storage requirements, and facilitating scalable execution on multi-billion point meshes [11].

The team is using TAU Commander and ParaTools ThreadSpotter to study and optimize computational performance of FDL3DI and AFSI on the latest Intel[®] Xeon Phi[™] architecture. By optimizing the time expensive kernels on Xeon Phi[™], these CFD codes may tackle bigger problems, reduce time to solution and utilize compute resources more efficiently. Since Intel[®]'s next generation of CPUs will also have an AVX-512 instruction set and architecture, these optimizations should be portable to future HPC platforms acquired at the Department of Defense (DoD). This work is conducted on the Intel[®] Xeon Phi[™] partition of Onyx, a Cray[®] XC40/50 with 544 Xeon Phi[™] 7230 compute nodes with 16GB of MCDRAM each and a Cray[®] Aries[™] interconnect. This system is located at the ERDC DoD supercomputing resource center.

We use the information provided by TAU Commander and ThreadSpotter to guide optimization efforts. TAU Commander can find source code regions that limit scalability using profiling, tracing, and aggregate summary statistics with respect to computational time, memory allocation, and hardware counter metrics. In addition, TAU Commander simplifies the TAU Performance System[®] [12], [13], and provides a local user database to organize performance characterization experiments. The TAU Performance System[®] has been used to improve the performance and stability of several DoD applications [14]–[17] in the past. TAU Commander reduces the number of commands a user must know from approximately eight to exactly one and reduces the number of unique steps in the performance workflow by approximately 50% [18]. Best

practices and workflows are being documented to assist other DoD groups in performance engineering efforts.

ParaTools ThreadSpotter analyzes an application as it is running and identifies memory access problems, loop structure problems and threading issues. It provides a high-level characterization of memory bandwidth, memory latency, data locality and thread communication and interaction issues on a per-MPI-rank basis. ThreadSpotter provides a report listing the detected inefficiencies in order of importance and suggests actionable remediations. Combined with TAU, it can target instrumentation to data movement hot spots.

First we performed a baseline characterization of FDL3DI and AFSI on the Xeon PhiTM partition Onyx using TAU Commander to identify regions that are the most time expensive, and exhibit large quantities of cache misses. Then we ran ThreadSpotter, focused on the regions previously identified by TAU Commander. Using recommendations from ThreadSpotter, optimizations are implemented and tested. The workflow was then repeated until the current source code region of interest is sufficiently optimized and the next problem area can be investigated.

The test problem for FDL3DI consists of a turbulent boundary layer, over a plate with a deformable section. We first found the optimal configuration for running FDL3DI on Xeon PhiTM nodes, which consisted of 64^3 blocks per MPI rank, with 8 threads per rank and two threads per physical core. This best-case baseline configuration is built upon, testing potential optimizations in this configuration. The test problem for AFSI is a data-parallel line relaxation (DPLR) implicit solver kernel mini-app. The DPLR implicit algorithm [19] is used in other codes, including NASA's DPLR CFD application [20] and US3D [21]. We tested different block sizes, 48^3 , 100^3 and 144^3 , which are the small, medium and large cases.

We investigated the effect of memory mode for both applications. Both achieved the best performance when the Xeon PhiTM node is configured in quadrant mode (rather than SNC4). All time expensive procedures in FDL3DI performed best when memory was allocated in high-bandwidth MCDRAM. A very small difference of just over 1% improvement was observed for FDL3DI when pinned to MCDRAM memory versus configuring the Xeon PhiTM in cache mode where 100% of MCDRAM acts as a cache. Conversely, for AFSI, the most time expensive procedure was fastest when allocated in DRAM, likely due to the very large cache miss rates observed, and DRAM's lower latency than MCDRAM.

The baseline analysis for both codes revealed that poor loop structure and cache utilization was degrading performance in the most time expensive procedures. For FDL3DI these were DERIV and TDGCOMP and for AFSI the slowest procedures were `factor` and `factor_stage_2`.

In FDL3DI we focused our attention on the slowest subroutine, TDGCOMP. TDGCOMP is a tridiagonal solver adapted from Numerical Recipes in Fortran [22]. This subroutine has an L2 cache miss rate of approximately 20%, and it grows steadily as the computation is scaled to more cores. In an attempt to increase data locality and reduce L1 and L2 cache

misses we performed loop interchange, loop fusion and loop blocking optimizations. We found 16 to be an optimal block size. This resulted in a reduction of L2 cache miss rate by over $\frac{2}{3}$ and a speedup of 3.75 for this subroutine.

Poor loop structure and data access patterns were present in AFSI's most time expensive subroutine, `factor_stage_2`. For *i*-line solves on the medium mesh, the L2 cache miss rate was over 95%. This may explain better performance of DRAM over MCDRAM, as discussed above. In addition, `factor_stage_2` is called regardless of the line-solve direction. The *k* direction is the most efficient. Geometries are meshed so that the wall-normal direction, and hence the line-solve direction, correspond with the *k* index if possible, resulting in unit stride access in memory. Despite this, cache miss rates were still higher than desired. For some geometries, having a mesh requiring *i*-line solves is unavoidable. This is the slowest line-solve direction and a real pain point for some simulations. *i*-line solves can only use value per cache-line and the next value is spatially distant in memory. ThreadSpotter identified multiple issues with `factor` and `factor_stage_2`. The issues included inefficient loop nesting, poor fetch utilization, and unpredictable memory access patterns. The suggested fixes included spatial and temporal loop blocking and loop fusion. To correct these issues, `factor_stage_2` was manually inlined into `factor` and specialized for solves in each of the line directions. Loop fusions were performed as well as a loop interchange and compiler loop blocking directives were employed. This led to a speedup of more than 9.5 in `factor` for *i*-line solves on the medium grid. L2 cache misses were reduced by an order of magnitude from $2.79E+09$ to $2.30E+08$. Inclusive time in `factor` dropped to under 10.5% of the baseline value.

For both applications, the biggest contributor to poor L2 cache utilization was the use of Fortran array assignments. Fortran semantics require the entire array assignment is performed before proceeding to the next statement. Multiple consecutive array assignment expressions that do not fit in cache will cause large arrays to be redundantly reloaded from DRAM or MCDRAM. Using smaller array statements in inner loops may be beneficial for vectorization. A loop over these blocked arrays may be interchanged to become an outer loop allowing for better data locality. Any remaining intermediate loops should be fused whenever possible.

ACKNOWLEDGMENTS

This material is based upon work supported by, or in part by, the Department of Defense High Performance Computing Modernization Program (HPCMP) under User Productivity, Technology Transfer and Training (PETTT) contract number GS04T09DBC0017 (PP-CFD-KY09-008-P3 and PP-CFD-KY09-001-P3 projects). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DoD HPCMP.

REFERENCES

- [1] D. V. Gaitonde and M. R. Visbal, "High-order schemes for navier-stokes equations: Algorithm implementation into fdl3di," Air Force Research Laboratory, Tech. Rep. AFRL-VA-WP-TR-1998-3060, August 1998.
- [2] N. J. Bisek, "High-fidelity numerical modeling of compressible flow," Air Force Research Laboratory, Tech. Rep. AFRL-RQ-WP-TR-2015-0160, November 2015.
- [3] M. R. Visbal and D. V. Gaitonde, "On the use of high-order finite-difference schemes on curvilinear and deforming meshes," *Journal of Computational Physics*, vol. 181, pp. 155–185, 2002.
- [4] Visbal, M. R. and Gaitonde, D. V., "Shock capturing using compact-differencing-based methods," in *AIAA Paper*, no. 2005-1265, 2005.
- [5] M. R. Visbal and D. P. Rizzetta, "Large eddy simulation on curvilinear grids using compact differencing and filtering schemes," *Journal of Fluids Engineering*, vol. 124, pp. 836–847, 2002.
- [6] Visbal, M. R., Morgan, P. E. and Rizzetta, D. P., "An implicit les approach based on high-order compact differencing and filtering schemes," in *AIAA Paper*, no. 2003-4098, 2003.
- [7] M. R. Visbal, "Analysis of the onset of dynamic stall using high-fidelity large-eddy simulations," in *52nd Aerospace Sciences Meeting*, 2014, p. 0591.
- [8] K. Bowcutt, A. Paull, D. Dolvin, and M. Smart, "Hifire: An international collaboration to advance the science and technology of hypersonic flight," in *Proceedings of the 28th International Congress of the Aeronautical Sciences*. ICAS Secretariat RA Leiden, The Netherlands, 2012, pp. 2012–998.
- [9] N. J. Bisek, "High-fidelity simulations of the hifire-6 flow path," in *54th AIAA Aerospace Sciences Meeting*, 2016, p. 1115.
- [10] R. Gosse, *Pyrolysis Gas Modeling of TPS Materials Using Finite-Volume Method and Issues with Validation*. American Institute of Aeronautics and Astronautics, 2018/07/11 2011. [Online]. Available: <https://doi.org/10.2514/6.2011-3641>
- [11] R. C. Gosse, C. Mackintosh, K. Vogiatzis, and C. Williams, *Parametric Domain Representation of Multi-Block Structured Meshes for Strong-Shock Flow Simulations*. American Institute of Aeronautics and Astronautics, 2018/07/11 2018. [Online]. Available: <https://doi.org/10.2514/6.2018-1404>
- [12] S. Shende and A. Malony, "The TAU Parallel Performance System," *International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.
- [13] A. Morris, A. Malony, S. Shende, and K. Huck, "Design and Implementation of a Hybrid Parallel Performance Measurement System," in *Proceedings of the International Conference on Parallel Processing (ICPP)*, 09 2010.
- [14] S. Shende, A. Malony, J. Linford, A. Wissink, and S. Adamec, "Isolating runtime faults with callstack debugging using tau," *Proc. HPEC 2012 Conference*, 2012.
- [15] S. Shende, A. D. Malony, A. Morris, and A. Wissink, "Simplifying memory, I/O, and communication performance assessment using TAU," in *Proceedings of DoD HPCMP UGC Conference*. IEEE Computer Society, 2010.
- [16] J. C. Linford, S. Shende, A. D. Malony, and A. Wissink, "Efficient Parallel Runtime Bounds Checking with the TAU Performance System," in *Proceedings of the 2013 IEEE High Performance Extreme Computing Conference (HPEC'13)*, Waltham, MA, 09 2013.
- [17] A. Wissink and S. Shende, "Performance Evaluation of the Multi-language Helios Rotorcraft Simulation Software," in *Proceedings of DoD HPCMP UGC Conference*. IEEE Computer Society, 2008.
- [18] J. C. Linford, S. Shende, and A. Malony, "Tau commander: An intuitive interface for the tau performance system," ParaTools, Inc., Chicago Operations Office, Argonne, IL, Announcement of USDOE Scientific and Technical Information OSTI ID 1252491, 04 2016.
- [19] M. J. Wright, G. V. Candler, and D. Bose, "Data-parallel line relaxation method for the navier-stokes equations," *AIAA journal*, vol. 36, no. 9, pp. 1603–1609, 1998.
- [20] M. J. Wright, T. White, and N. Mangini, *Data Parallel Line Relaxation (DPLR) Code User Manual: Acadia-Version 4.01. 1*, NASA, 2009.
- [21] I. Nompelis, T. Drayna, and G. Candler, "Development of a hybrid unstructured implicit solver for the simulation of reacting flows over complex geometries," in *34th AIAA Fluid Dynamics Conference and Exhibit*, 2004, p. 2227.
- [22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in FORTRAN 77, vol. 1*. New York, NY: Press Syndicate of the University of Cambridge, 1992.