Replicated Computational Results (RCR) Report for "BLIS: A Framework for Rapidly Instantiating BLAS Functionality"

JAMES M. WILLENBRING, Sandia National Laboratories

"BLIS: A Framework for Rapidly Instantiating BLAS Functionality" by Field G. Van Zee and Robert A. van de Geijn (see: http://dx.doi.org/10.1145/2764454) includes single-platform BLIS performance results for both level-2 and level-3 operations that is competitive with OpenBLAS, ATLAS, and Intel MKL. A detailed description of the configuration used to generate the performance results was provided to the reviewer by the authors. All the software components used in the comparison were reinstalled and new performance results were generated and compared to the original results. After completing this process, the published results are deemed replicable by the reviewer.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: *Efficiency*

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Linear algebra, libraries, high-performance, matirx, BLAS, Replicated Computational Results

ACM Reference Format:
James M. Willenbring, 2015. Replicated computational results (RCR) report for "BLIS: A framework for rapidly instantiating BLAS functionality". ACM Trans. Math. Softw. 41, 3, Article 15 (May 2015), 4 pages.
DOI: http://dx.doi.org/10.1145/2738033

1. INTRODUCTION

The results replication effort for BLIS: A Framework for Rapidly Instantiating BLAS Functionality was focused on Section 7 of the manuscript, which provides performance comparisons for a number of level-2 and level-3 BLIS operations against BLAS operations in the MKL, ATLAS, and OpenBLAS libraries. The authors granted the reviewer access to the machine (described in Section 7.1) on which the results were generated. This machine was also used to generate all of the replicated results.

2. REPLICATING THE RESULTS

The RCR process consisted of installing the same four libraries used to produce the original performance results:

—MKL 11.0 Update 4,
—ATLAS 3.10.1,
—OpenBLAS 0.2.6,
—BLIS 0.1.0-20.

Then, we compiled and ran the BLIS test drivers, which generated performance results for all tested libraries.

Author's address: Center for Computing Research, Sandia National Laboratories, Albuquerque, NM 87123; email: jmwille@sandia.gov.

0098-3500/2015/05-ART15 $15.00
DOI: http://dx.doi.org/10.1145/2738033

2.1. Installation

The four codes used to generate replicated computational results, MKL, ATLAS, OpenBLAS, and BLIS, were installed by the reviewer using instructions provided by the authors. Using new rather than existing installations of all four codes provides greater clarity for the replication process. For all commands shown in this article, replace the text inside brackets with the appropriate path (<path>).

GCC 4.8.2 was used where compiling was necessary. This version was custom installed using the instructions on the GCC wiki [Wakely 2013].

Installing MKL Version 11.0, Update 4 required no configuration. The only requirements were a copy of the code, and a license to point to during the installation process.

To install ATLAS 3.10.1, the following configuration command was used from a build directory that is a child of the top-level ATLAS directory.

```
../configure --prefix=<atlas_installation_path> \
--with-netlib-lapack-tarfile=<lapack_tar_path>/lapack-3.3.1.tgz \
-b 64 -t 0 -m 2666 -Si latune 0
```

A full discussion of the configuration options used is beyond the scope of this review, we note however that threading is disabled. Also, LAPACK is not used to generate the performance results; using an older version of LAPACK for the configuration is functionally equivalent for this effort. The command `../configure --help` provides a full list of configuration options with a short description.

When compiling ATLAS, the following error was encountered.

```
ar: cblas_dptgemm.o: File format not recognized.
```

The error appears to be related to the no threads option, and did not prevent any necessary part of ATLAS from building. After consulting with the authors, it was decided not to investigate the error further.

OpenBLAS 0.2.6 was configured by modifying the `Makefile.rule` to enable 64-bit object code, turn off threading, and exclude CBLAS, LAPACK, and LAPACKE.

The source code for BLIS was obtained directly from the BLIS git repository [Van Zee 2015]. Initially, there was no obvious way to identify the version of BLIS the authors used to generate their results. The reviewer offered a couple of possible solutions. The authors graciously chose the recommendation that provided the greatest clarity and replicability and took the time to regenerate their results based on a specific version and commit. The results were generated based on commit `f60c8adc2f61`, which corresponds to version 0.1.0-20 (dated December 10, 2013). BLIS was configured using the following.

```
./configure dunnington p <BLIS_install_prefix>
```

2.2. Replicating Level-2 and Level-3 Cross-Library Performance Comparisons

All of the Level-2 and Level-3 cross-library performance results are generated using the BLIS test driver. After installing BLIS, before running the test driver, it is necessary to first edit the `Makefile` in the `blis/test` directory to point to the installations of MKL, ATLAS, OpenBLAS, BLIS, and GCC (`LDFLAGS` only). It was also necessary to set `LD_LIBRARY_PATH` to find the MKL libraries in `<MKL_install_path>/intel/mkl/lib/intel64`.

The `runme.sh` script in the same directory will generate the results shown in Figures 13 and 14. The output of the script is stored in Matlab `.m` files. Performance

graphs can be generated from the result files using scripts available from the ACM Digital Library, or from the authors upon request.

2.3. Replicating TRSM-Variant Performance Comparisons

Replicating the results in Figure 15 required small manual changes to BLIS for the unfused unrolled trsm, SSE and unfused generic trsm, FPU results. The fused gemmtrsm results were produced using the process described in Section 2.2. Each set of manual changes was made to select the proper kernels and flags for the variant being tested. Then an executable was built (named `test_trsm_blis.x` for all variants) and used to produce output files named based on the variant being tested.

The unfused unrolled trsm, SSE results were replicated by first modifying `blis/config/dunnington/bli_kernel.h` in the "LEVEL-3 KERNEL DEFINITIONS" section to include the lines pertaining to the gemmtrsm ref kernel, and comment out the corresponding opt kernel lines. Similarly, the unrolled ref 4×4 trsm loop was included, and the loop-based ref kernel was commented out. Next, a symlink was needed in the `dunnington` directory to find all necessary kernels.

```
ln -s ../../kernels/c99 kernels_c99
```

Then BLIS had to be reconfigured and rebuilt, and the test driver had to be rebuilt. Finally, the results were created with the following command.

```
./test_trsm_blis.x > output_trsm_blis_nofus_sse_4x4.m.
```

The unfused generic trsm, FPU results required `bli_kernel.h` to be modified to switch back to the loop-based trsm kernel rather than the unrolled kernel used previously. Also, in the definition of `CKOPTFLAGS` in `blis/config/dunnington/make_defs.mk` was updated to include `-mfpmath=387` instead of `-mfpmath=sse` (recall that the previous results were for the unfused unrolled trsm, SSE variant and that the flags chosen were selected to optimize for each variant). Then BLIS was reconfigured, and rebuilt, the test driver was rebuilt, and the final results were generated using the following.

```
./test_trsm_blis.x > output_trsm_blis_nofus_fpu_mxn.m
```

Prior to plotting the results for Figure 15, minor changes were necessary to the beginning of each line in the `.m` files to match the labels used in the graphs. This could have been avoided by changing the label in the test code prior to generating the result.

3. EVALUATION OF REPLICATED RESULTS

When evaluating whether or not the results in the BLIS article were successfully replicated as described previously, it is important to reiterate the claim made by the authors with regards to performance. Specifically, the following was listed as the fourth contribution of the article in Section 1.

> It demonstrates level-2 and level-3 performance that is highly competitive with (and in some cases, superior to) existing BLAS implementations, such as OpenBLAS, ATLAS, and Intel's Math Kernel Library (MKL).

Since no claims were made specifically about relative performance, or speedups, the reviewer did not deem it necessary to compute any performance-related numbers. Rather, all of the graphs generated by the reviewer were carefully visually compared to the graphs generated by the authors.

Based on this comparison, the reviewer deems the results to be successfully replicated. Although there were slight differences in the results, as one would expect for computational results of this nature, the differences do not invalidate the claim that BLIS is highly competitive with, and in some cases superior to the other BLAS implementations used in the article.

The differences that were present were not consistently biased in favor of or against any BLAS implementation. The overall relative performance of the implementations for each replicated graph was consistent with the published graph. For Figure 15, which is not specifically related to the contribution cited previously, the graphs were very similar as well and consistent in relative performance. Our versions of these graphs are available in the ACM Digital Library.

4. CONCLUDING REMARKS

The scope of this review was limited to replicating the specific results obtained by the authors on a specific platform, using a specific configuration for each piece of software. However, the reviewer did not note any obvious inherent impediment to attempting to run a similar analysis elsewhere, or experimenting with different configuration parameters where appropriate. As described, replicating the results for Figures 13 and 14 was simpler than for Figure 15, but those results can also be generated with a little bit of care. All of the required software is freely and openly available.[1]

The results published in the BLIS article are deemed replicable by this reviewer.

ACKNOWLEDGMENTS

The reviewer would like to thank Field Van Zee for providing all of the details necessary to complete the review. Sincere thanks also go out to Michael Heroux for several rounds of feedback and guidance and to Ronald Boisvert and Juliana Freire for their insights on results replication reports.

REFERENCES

Field G. Van Zee. 2015. BLIS BLAS-like library instantiation software. https://github.com/flame/blis.
Jonathan Wakely. 2013. Installing GCC. http://gcc.gnu.org/wiki/InstallingGCC.

Received October 2014; accepted February 2015

[1]A free 30-day evaluation of MKL can be downloaded from Intel's website.