Choosing the optimal BLAS and LAPACK library

Tobias Wittwer
Version 1.0

March 14, 2008

Delft Institute of Earth Observation and Space Systems (DEOS)
Delft University of Technology, Kluyverweg 1, 2629HS Delft, The Netherlands
Tel.: +31-15-2788146, Fax: +31-15-2782348, t.f.wittwer@tudelft.nl

Abstract

Most geodetic applications require the assembly and solution of linear equation systems. The BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra Package) have established itself as quasi-standard for linear algebra computational routines. The performance that can be achieved with a program making use of these routines is largely dependent on the performance of the chosen BLAS and LAPACK library.

This article investigates the performance of several BLAS and LAPACK implementations on three common PC architectures. A number of test computations have been made, using a program for spherical harmonic analysis with least squares. Based on the results, recommendations for the optimal library for each architecture are made.

Keywords: BLAS, LAPACK, spherical harmonic analysis

1 Introduction

Most geodetic problems require a parameter estimation process. A current research field that involves the estimation of certain parameters out of a large number of observations are the gravity field recovery from satellite missions such as CHAMP, GRACE, and in the future, GOCE, see eg. [Ditmar et al. 2002]. Another example is the computation of the terrestrial reference frame out of GPS, SLR, VLBI and DORIS observations, described by [Altamimi et al. 2002].

A linear equation system of the structure

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{1}$$

is commonly solved using least-squares estimation:

$$\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} = \mathbf{N}^{-1}\mathbf{b}, \tag{2}$$

where $\mathbf{y}$ are the observations and $\hat{\mathbf{x}}$ the estimated unknown parameters. $\mathbf{A}$ is commonly referred to as design matrix, $\mathbf{N}$ as normal equation matrix, and $\mathbf{b}$ as right-hand-side vector.

The Basic Linear Algebra Subprograms (BLAS) are a standard for linear algebra routines. They are sorted into level 1 (scalar-vector and vector-vector operations), level 2 (matrix-vector operations), and level 3 (matrix-matrix operations) routines. For example, the DGEMM routine is a double-precision (D) dense (GE - general) matrix-matrix multiplication. It, or the DSYRK routine, which performs a matrix-matrix multiplication $\mathbf{B} = \mathbf{A}^T\mathbf{A}$ or $\mathbf{B} = \mathbf{A}\mathbf{A}^T$, are usually used for computing the normal equation matrix $\mathbf{N}$ in equation 2.

The Linear Algebra Package (LAPACK) is a standard for routines such as solving of linear equation systems, LU, QR, and singular value decomposition. The solution of the positive-definite linear equation system in equation 2 can be computed with the DPOSV routine (double-precision positive-definite solver).

Note that both BLAS and LAPACK are only standards, defining functionality and interfaces. The actual implementation is not standardised. Over the years, many implementations for different architectures or using different approaches have been created. In many parameter estimation programs, most of the computational load will be performed by BLAS and LAPACK routines. It is thus desirable to use the implementations that offer the best performance, in order to reduce computation times.

This article has the objective of identifying the BLAS and LAPACK libraries that offer the best performance on current popular computer architectures. To achieve this goal, a number of test computations were made, using three different computer systems and five different BLAS and LAPACK implementations. All tests computations were made

using a program for spherical harmonic analysis (SHA) using least squares, implementing equation 2. Based on the results, a recommendation for the optimal BLAS and LAPACK implementation can be made.

The paper is organized as follows: Section 2 describes the methods used to assess the performance of the tested BLAS and LAPACK libraries. Section 3 described the hardware and software used for the tests. In section 4, the results of the computations are presented and discussed. Finally, in section 5 a summary is provided and conclusions are drawn.

2 Performance Assessment

As described in the previous section, most of the workload in a parameter estimation according to equation 2 is contained in the dense matrix-matrix multiplication

$$\mathbf{N} = \mathbf{A}^T\mathbf{A}, \tag{3}$$

and the solving of the linear equation system

$$\mathbf{b} = \mathbf{N}\mathbf{x}. \tag{4}$$

The operation in equation 3 is performed by the DGEMM or DSYRK routine of BLAS, the linear equation system in equation 4 is solved by the DPOSV routine. Performance comparisons can be made by measuring the time spent in these functions. A system-independent routine for measuring time is the OMP_GET_WTIME routine of OpenMP, a standard for parallel programming on shared-memory computers ([OpenMP 2002]). A call to OMP_GET_WTIME returns a time in seconds, the difference between two calls yields the time elapsed between them.

While it is sufficient to compare different implementations based on their runtimes, it is desirable to be able to compare the theoretical achievable and actually achieved performance. The ratio between achieved performance R and theoretical peak performance R_{peak} is known as efficiency. A high efficiency is an indication of an efficient numerical implementation.

Performance is measured in floating point operations per second, FLOPS, or FLOP/s. Current processors deliver an R_{peak} in the GFLOPS (10^9 FLOPS) range. The $R_{peak}$of a system can be computed by

$$R_{peak} = n_{CPU} \cdot n_{core} \cdot n_{FPU} \cdot f, \tag{5}$$

where n_{CPU} is the number of CPUs in the system, n_{core} is the number of computing cores per CPU, $n_{FPU}$is the number of floating point units per core, and f is the clock frequency.

The achieved performance R can be computed as the number of floating point operations performed (known as flop-count) divided by the time taken. For the DGEMM routine of BLAS, the number of floating-point operations is

$$n_{flop} = 2mnk, \tag{6}$$

with m being the number of rows of the first matrix, n being the number of columns of the second matrix, and k being the number of columns of the first matrix and the number of rows of the second matrix. The flop count of the DSYRK routine is

$$n_{flop} = m(m+1)n, \tag{7}$$

with n being the number of rows of $\mathbf{A}$, and m being the number of columns of $\mathbf{A}$ and the number of rows and columns of $\mathbf{N}$.

All tests were done using SHALE, a program for spherical harmonic analysis using least squares. SHALE and the parallelisation of it are described in great detail in [Wittwer 2006]. SHALE estimates spherical harmonic potential coefficients out of gravity potential values. The estimation is performed as in equation 2, using the DGEMM/DSYRK and DPOSV routines of BLAS and LAPACK. SHALE is available for download from the author's website, `http://www.lr.tudelft.nl/psg` → Staff → Tobias Wittwer → personal homepage. The time spent by the DGEMM routine, as well as the performance and efficiency achieved by it, and the time spent by the DPOSV routine, will be used to judge the performance of the various BLAS and LAPACK implementations. DGEMM performance is usually used as benchmark for BLAS performance, as other BLAS routines (such as DSYRK) also make use of DGEMM.

3 Test Setups

The goal of the tests was to find the best BLAS and LAPACK libraries for the most popular computer architectures in use. The last years have seen an almost complete move away from RISC-based workstations and supercomputers to personal computer (PC) architectures. The ever-increasing performance demands of PC software, as well as the race between the two manufacturers AMD and Intel, have led to a performance gain that the traditional architectures for numerical computing could not follow. Even the high performance computing field now mostly uses x86-family CPUs, due to their lower costs and higher performance when compared to RISC processor families such as MIPS, UltraSPARC and Alpha.

system	n_{CPU}	n_{core}	n_{FPU}	f	R_{peak}
Opteron	2	2	3	2.4 GHz	28.8 GFLOPS
Pentium D	1	2	2	3 GHz	12 GFLOPS
Core	1	2	4	2.4 GHz	19.2 GFLOPS

Table 1: Theoretical peak performance R_{peak} of the systems tested, computed according to eq. 5.

As a consequence, only x86-family CPUs were used in the tests described in this article. There are today two manufacturers controlling the x86 processor market, AMD and Intel. Both offer an almost endless variety of processor models, from low-voltage CPUs for notebook computers over low-cost models for office applications to multi-CPU capable designs for servers and "Extreme"-dubbed high-end processors for gamers.

This huge array of choices can be narrowed down to three architectures that are of importance in the field of numerical computing:

- AMD's "Hammer" architecure, so called of the codenames "Sledgehammer" and "Clawhammer" of its first models, the official designation is "K8". These are AMD's 64-bit processors. They are available as Athlon aimed at the consumer market, and Opteron aimed at the workstation and server market. Only Opterons are available in multi-CPU capable versions.
- Intel's "Netburst" architecture, as used in all Pentium 4 processors. The Netburst architecture has been designed for high CPU clock speeds. Even though this architecture is not further developed, Netburst-based CPUs are still produced and still offer the highest clock frequencies available. The multi-CPU capable derivatives are called Xeon.
- Intel's "Core" architecture, the replacement of the Netburst architecture. The Core architecture has been developed out of the successful Pentium M design, with ancestry dating back to the Pentium Pro. While featuring lower clock frequencies than Netburst-based CPUs, they are known for their high efficiency. Once again, the multi-CPU capable versions, aimed at the server and workstation market, are sold under the Xeon name.

For each architecture, one representative CPU was chosen. Because of their performance benefit, only dual-core CPUs were taken into consideration. The final choice was:

- A system equipped with two AMD Opteron 280 CPUs. The Opteron 280 is a dual-core CPU clocked at 2.4 GHz. Each core has 1 MB of L2 cache. This system is referred to as the Opteron system.
- A system equipped with an Intel Pentium D 830 CPU. The Pentium D is the dual-core version of the Pentium 4. The Pentium D 830 is clocked at 3 GHz and has 1 MB of L2 cache per core. This system is referred to as the Pentium D system.
- A system equipped with an Intel Core 2 Duo E6600 CPU. This dual-core representative of the Core architecture is clocked at 2.4 GHz and has 4 MB of L2 cache. This system is referred to as the Core system.

All CPUs are 64-bit capable and were run with 64-bit Linux operating systems, as well as 64-bit compilers. The Intel C/C++ (icc) and Fortran (ifort) compilers in version 10.0.025 were used. These are known to generate very efficient code on both AMD and Intel processors. All libraries were compiled with -O3 and -xW (for AMD) or -xP (for Intel) optimisation switches.

Five common and well-known BLAS and LAPACK implementations were tested:

- The reference BLAS and LAPACK (in version 3.1.1) libraries are reference implementations of the BLAS [Lawson et al. 1979] and LAPACK [Anderson et al. 1999] standard. These are not optimised and not multi-threaded, so not much performance should be expected. These libraries are available for download at `http://www.netlib.org/blas` and `http://www.netlib.org/lapack`.
- The Automatically Tuned Linear Algebra Software, ATLAS [Whaley et al. 2005], in version 3.6.0. During compile time, ATLAS automatically choses the algorithms delivering the best performance. ATLAS does not contain all LAPACK functionality; it can be downloaded from `http://www.netlib.org/atlas`.
- The Goto BLAS in version 1.15, an implementation of the level 3 BLAS aimed at high efficiency [Goto et al. 2006]. Since this is a BLAS-only library, the LAPACK reference implementation was used with it. The Goto BLAS is available for download from `http://www.tacc.utexas.edu/resources/software`.
- The AMD Core Math Library (ACML) in version 3.6.0. This is AMD's implementation of the BLAS and LAPACK standards. Other functionality is offered as well, such as vectorised mathematical routines. The ACML is available at `http://developer.amd.com/acml.jsp`.
- The Intel Math Kernel Library (MKL) in version 9.1.021. Intel's implementation of the BLAS and LAPACK standard provides further functionality, such as fast-fourier transform (FFT) and vectorised mathematical routines. A version for non-commercial use can be downloaded from Intel's website, `http://www.intel.com`.

4 Results and Discussions

All tests were performed with SHALE, computing a spherical harmonic analysis up to degree 50, which results in 2,601 unknown potential coefficients. 16,200 potential values were used as observations.

The first test was to assess the singlethreaded DGEMM performance of the five libraries. Singlethreaded performance is of interest if either only single-processor, single-core systems are used, or if parallelism is achieved by some other method than multithreading, such as message with with MPI or PVM. For the Goto BLAS and the MKL, the number of threads was manually set to 1. As ATLAS and the ACML provide singlethreaded libraries, these were used. The reference BLAS is not threaded, so no special precautions had to be taken.

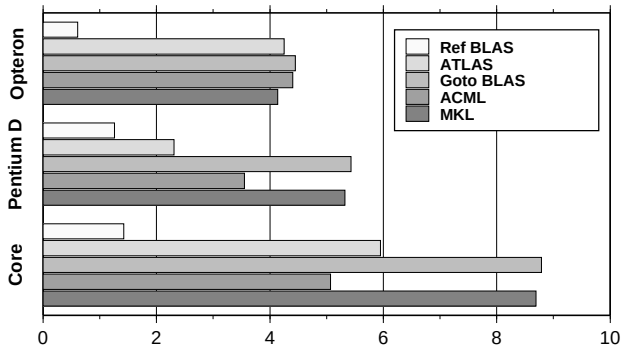

Figure 1: Singlethreaded DGEMM performance in GFLOPS for Opteron (top), Pentium D (middle), and Core (bottom) systems.

Figure 1 shows the resulting performance for all three systems and all five libraries. As expected, the non-optimised reference BLAS implementation offers only poor performance. The best performance is achieved by the Goto BLAS, on all three systems. The vendor-specific libraries (ACML for the Opteron system, MKL for the Pentium D and Core system) are only slightly slower. Both ATLAS and MKL deliver good performance on the Opteron system. On the Intel systems though, ATLAS and ACML deliver significantly less performance than Goto BLAS and MKL.

The second test aimed at testing multithreaded DGEMM performance. The maximum sensible number of threads (four on the Opteron system, two on the two Intel systems) were used, with multithreaded version of all libraries. The reference BLAS results will not differ from those obtained in the previous test, as the reference BLAS is not parallelised.

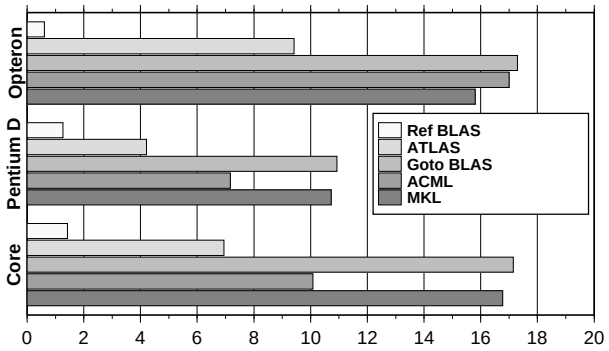

Figure 2: Multithreaded DGEMM performance in GFLOPS for Opteron (top), Pentium D (middle), and Core (bottom) systems.

The results are shown in figure 2. They differ little from the previous test.Once again, the Goto BLAS is the fastest, with the respective vendor libraries a close second. Compared to the singlethreaded tests, ATLAS performs less well. The multithreaded implementation does seem to be less efficient than the singlethreaded library.

Figure 3 shows the corresponding efficiencies. Both Intel systems achieve efficiencies close to 90% with the Goto BLAS and the MKL. The Opteron system's efficiency is around 60% - a result of the Opteron processor design. It has only one load/store unit, while the Core 2 has a dedicated load and store unit each.

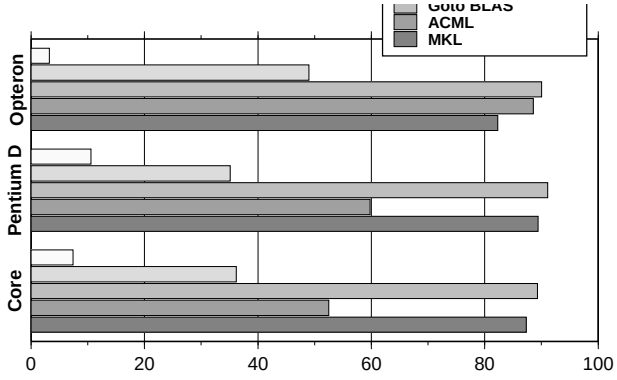

Figure 3: Multithreaded DGEMM efficiency in % for Opteron (top), Pentium D (middle), and Core (bottom) systems.

The final test was run to compare the performance of the various LAPACK implementations. Note that the Goto BLAS does not contain LAPACK routines, so it was used together with the reference LAPACK. LAPACK performance is mostly dependent on the performance of the underlying BLAS implementation. The linear equation system with 2601 unknowns was solved using LAPACK's DPOSV routine. Figure 4 shows the resulting runtimes.

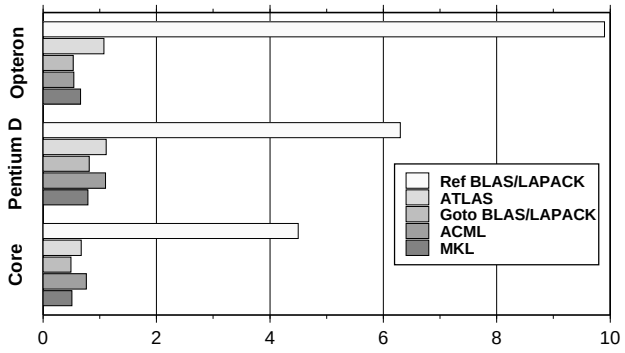

Figure 4: Multithreaded DPOSV runtimes in seconds for Opteron (top), Pentium D (middle), and Core (bottom) systems.

The reference BLAS/LAPACK combination clearly requires the most runtime, a result that's not surprising considering LAPACK performance depends on BLAS performance. The Goto BLAS/reference LAPACK combination is the fastest on all systems, closely followed by the vendor specific libraries (ACML on AMD, MKL on Intel). ATLAS is slightly slower, and once again MKL fares better on the Opteron system than ACML on the Intel systems.

5 Summary and Conclusions

The goal of the tests described in this article was to identify the optimal BLAS and LAPACK libraries for the most important PC architectures, AMD Hammer, Intel Netburst, and Intel Core. Performance was assessed using the DGEMM routine for dense matrix-matrix multiplication, and the DPOSV routine for the solving of linear equation systems.

The results clearly showed that the Goto BLAS offers the best performance on all systems. It can be combined with the reference LAPACK implementation to provide a very fast BLAS/LAPACK library. The respective vendor libraries (AMD ACML and Intel MKL) deliver only slightly less performance. They are very good choices when their additional functionality (vectorised math functions, FFT) is required. The ATLAS could only deliver comparable performance in the singlethreaded test run on the Opteron system, but is otherwise significantly slower. Use of the reference BLAS should be avoided, as it is not optimised and thus delivers very poor performance.

References

[Altamimi et al. 2002] Altamimi Z, Sillard P, Boucher C (2002) ITRF2000: A new release of the International Terrestrial Reference Frame for earth science applications, J Geophys Res 107(B10): 2214, DOI 10.1029/2001JB000561

[Anderson et al. 1999] Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D (1999) LAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA.

[Ditmar et al. 2002] Ditmar P, Klees R (2002) A method to compute the Earth's gravity field from SGG/SST data to be acquired by the GOCE satellite. Delft University Press (DUP) Science, Delft.

[Goto et al. 2006] Goto K, van de Geijn R (2006) High-Performance Implementation of the Level-3 BLAS. ACM Trans Math Soft, submitted.

[Lawson et al. 1979] Lawson CL, Hanson RJ, Kincaid D, Krogh FT (1979) Basic Linear Algebra Subprograms for FORTRAN usage. ACM Trans Math Soft 5, 308-323.

[OpenMP 2002] OpenMP Fortran Application Program Interface, Version 2.0, 2002. http://www.openmp.org.

[Whaley et al. 2005] Whaley RC, Petitet A (2005) Minimizing development and maintenance costs in supporting persistently optimized BLAS. Software: Practice and Experience 35 (2), 101-121.

[Wittwer 2006] Wittwer T (2006) An Introduction to Parallel Programming. VSSD uitgeverij, Delft, ISBN 90-71301-78-8.