From Parallel Programming and Applications, P. Fritzon and L. Finmo (eds), IOS Press, 1995, pp 184–188.

GEMM–Based Level 3 BLAS: High–Performance Model Implementations and Performance Evaluation Benchmark

Bo Kågström, Per Ling, Charles Van Loan *

Department of Computing Science

Umeå University

S-901 87 Umeå, Sweden

Extended Abstract

1 Introduction

The memory organization in current advanced computer architectures is hierarchical. Accesses to data in the upper levels of the memory hierarchy (registers, cache and/or local memory) are much faster than those in lower levels (off–processor and shared memory). Typically, the peak performance measured in Mflops or Gflops (10^6 and 10^9 floating point operations per second, respectively) is only delivered for data stored in the top level of the memory hierarchy. Therefore, it is important to organize the computations such that we can maximize reuse of data in the upper levels of the memory hierarchy. Matrix and vector operations are basic in most scientific computations and can most often be reorganized for effective data reuse. For example, an n-by-n matrix multiply $C = AB$ involves $O(n^3)$ arithmetic but $O(n^2)$ data movements (usually). Thus, as n grows the cost of accessing data is increasingly dominated by the cost of computation. This fact has lead to a technique to reorganize standard algorithms to perform matrix–matrix (level 3) operations in their inner loops. Typically, these matrix–matrix operations are expressed as calls to level 3 Basic Linear Algebra Subprograms (BLAS) [4, 5], which together with level 1 and level 2 BLAS are defacto standards for basic matrix and vector operations. The level 3 BLAS have been succesfully used as building blocks for several applications, including the software library LAPACK [1].

With a highly optimized level 3 BLAS, most of the LAPACK codes will "automatically" peform well. However, due to the complex hardware organization of advanced computer architectures it can be very costly and time consuming to develop a high–performance level 3 BLAS because it requires assembly level programming/thinking. The GEMM–based approach presents a way to attain high performance and portability with a limited effort. Often computer systems have efficient implementations of the general matrix multiply operation. The GEMM–based level 3 BLAS concept [13] shows that it is possible to formulate the level 3 BLAS operations in terms of the level 3 operation for general matrix multiply and add (GEMM) and some level 1 and level 2 BLAS operations.

*Department of Computer Science, Cornell University, Ithaca, New York 14853-7501

We are in the process of finishing a two-part paper that will be submitted to ACM Transactions on Mathematical Software [12, 11]. The papers present portable and high–performance model implementations of the GEMM–based level 3 BLAS in Fortran 77 and the GEMM–based level 3 BLAS benchmark, which is a tool for performance evaluation of different level 3 BLAS implementations. Some early results from the model implementations have been published in [8, 9]. In this contribution (talk) we will discuss design principles for the model implementations and present new performance results for different architectures (vector as well as RISC–based), including single processor results for IBM SP2, Intel Paragon, Parsytec GC/PowerPlus and Silicon Graphics. Notice that any implementation of distributed versions of level 3 operations or block–partitioned algorithms must be designed to minimize communication overhead, as well as to make use of highly optimized single processor level 3 kernels. For an increasing number of processors, any improvement of the performance of a single processor will (at least for a single program multiple data (SPMD) application) have a multiplicative effect on the overall performance. Morover, we will present the GEMM–based level 3 BLAS benchmark, its purpose and design, and some benchmark results for different level 3 BLAS implementations.

2 The GEMM–based level 3 BLAS model implementations

The Level 3 BLAS consists of routines for both general and "structured" matrix multiplication. Multiple right hand side triangular system solving is also handled by the package as it is rich in matrix multiplication if properly organized.

We have shown that "one can live with" just one highly optimized Level 3 BLAS routine: _GEMM [13, 8]. This subprogram oversees a general matrix multiply of the form $C \leftarrow \alpha \mathrm{op}(A)\mathrm{op}(B) + \beta C$, where $\mathrm{op}(X)$ denotes X or X^T. The structured matrix multiplication problems handled by the other Level 3 BLAS can be couched in terms of _GEMM and a negligible amount of Level 1 and 2 computations, with suitable partitionings. Roughly, the idea is to reduce the overall structured multiplication to a set of general multiplications involving "strips". Here, a strip is either a block row or a block column. For performance purposes a strip can be further partitioned into subblocks.

GEMM–Based Level 3 BLAS can be implemented in different ways. Performance can be significantly affected by different design decisions, even if the underlying BLAS routines are well optimized and show high performance.

The model implementations are written in Fortran 77 and designed to be highly efficient on machines with a memory hierarchy. All routines are effectively structured to reduce data traffic in a memory hierarchy. A detailed description of the algorithms used in our model implementation for the different level 3 operations is presented in [10]. The design principles, including blocking strategy, use of local arrays and alternative code sections performing the same task are discussed in [10] (and to some extent in [8]). It is beyond the scope of this extended abstract to present any design principles in detail.

The user supplies underlying routines, the level 3 BLAS routine _GEMM and some level 1 and level 2 BLAS kernels. If they are efficiently optimized for the target machine, the GEMM–based level 3 BLAS model implementations offer:

- Efficient use of vector instructions (compound instructions, chaining, etc.), through _GEMM, level 1 and level 2 BLAS routines.
- Vector register reuse, through _GEMM and level 2 BLAS routines.

- Efficient cache reuse, through internal blocking, use of local arrays, and through _GEMM.

- Column–wise referencing, for problems that would cause severe performance degradation with row–wise referencing (except for reference patterns in underlying BLAS routines).

- Parallelism, through automatic parallelization by a compiler, or by using parallel underlying BLAS kernels.

- A level 3 BLAS library based on unconventional underlying matrix multiply algorithms like, for example, Strassen's or Winograd's algorithms.

3 The GEMM–based level 3 BLAS benchmark

Many people have put a lot of effort into developing fast level 3 BLAS since the specification was published in 1990 [4, 5]. Some vendors provide highly optimized BLAS for their machines, see for example [7, 2], while others provide optimized versions of some or none of the routines. Independent groups have also developed tuned level 3 kernels for different machines, for example [14, 6]. Today different implementations with different performance characteristics coexist and it is becoming more important to evaluate different implementations thoroughly.

The GEMM–based benchmark measures the performance of an arbitrary set of level 3 BLAS implementations, specified by the user, and compares it with the performance of the GEMM–based level 3 BLAS model implementations, permanently included in the benchmark. The level 3 BLAS implementations specified by the user are linked with the benchmark program. When the benchmark executes, timings are performed according to specifications given in an input file. The user may design her/his own tests or use the enclosed input files. The following output results are eligible for presentation:

- A collected "mean value" statistic, calculated from the performance results of the user specified level 3 BLAS routines for the problem configurations specified in the input file.

- Tables, that show performance results in Mflops for each routine and choice of parameters. Both the user–specified and the GEMM–based level 3 BLAS routines are timed and their performances are compared.

The tables are intended for program developers and others who require detailed performance information. It is possible to choose which of the following results that are to be presented:

- Performance of the GEMM–based level 3 BLAS routines in Mflops.

- Performance of the user–specified level 3 BLAS routines in Mflops.

- Performance of the user–specified _GEMM routine in Mflops.

- GEMM–efficiency of the user–specified level 3 BLAS routines.

- GEMM–ratio.

Results are presented for each routine and problem configuration specified in the input file. Each item in the listing above corresponds to a column of the output tables. The last two items are defined as follows. The GEMM–efficiency is intended to illustrate how close to the "practical" peak performance a routine reaches, which is defined as the performance of the highly optimized _GEMM routine specified by the user for the given problem configuration. Notice that the "practical" peak performance can be considerably lower than the maximal theoretical performance of the architecture considered. The performance of the user–specified level 3 BLAS routine is compared with the performance of the user–specified _GEMM routine. Let *Perf*(x) denote the the performance in Mflops of x. Then

$$\text{GEMM–efficiency} = \frac{\textit{Perf}(\text{user–specified level 3 routine})}{\textit{Perf}(\text{user–specified \_GEMM routine})}$$

The GEMM–efficiency is measured using a choice of parameters for _GEMM which, in this respect, "corresponds" to the problem configuration for the level 3 routine it is compared with.

The GEMM–ratio is the performance of a GEMM–based level 3 BLAS routine compared with the user–specified implementation of the same routine, i.e.,

$$\text{GEMM–ratio} = \frac{\textit{Perf}(\text{GEMM–based routine})}{\textit{Perf}(\text{user–specified level 3 routine})}$$

For a vendor supplied level 3 BLAS library we would expect to have all GEMM–ratios less than 1. However, this is not always the case (e.g. see results in [8, 9]). A value greater than one implies that the GEMM–based implementation is faster than the user–specified implementation for the given problem configuration.

The collected "mean value" statistic provide a comprehensive performance result of the user–supplied routines that is easy to compare with other level 3 BLAS implementations, and between different machines. This result consists of a tuple (x, y), where x is the mean value of the GEMM–efficiency and y is the mean value of the performance of _GEMM. The GEMM–efficiency is measured for the routines and choice of parameters that are specified in the input file and the performance of _GEMM is measured for corresponding problem configurations. Provided that _GEMM is well implemented, y represents the average "practical" peak performance of the machine, for the specified problem configurations. The average performance of the remaning user–specified implementations can be approximated as $x \cdot 100$ percent of the average "practical" peak performance y.

To further standardize the result we have included a pair of "canonical" input files called _MARK01 and _MARK02. The user can also use "customized" input files to obtain results for problem configurations of interest.

4 Availability and installation

The source code for the model implementations and for the benchmark program comes in single, double, complex, and double complex precision data types and is freely available via netlib [3]. All routines are written in Fortran 77 for portability. No changes should be necessary to run the programs correctly on different target machines. Send an e–mail with the following message

send index
send index from blas/gemm_based

to *netlib@research.att.com*, for further information on how to obtain the programs, or use the World Wide Web

http://www.netlib.org/blas/gemm_based/

to obtain the programs. All programs are also available via anonymous ftp from *ftp.cs.umu.se*. Get the file gemm_based.tar.Z under the directory pub.

Both the model implementations and the benchmark come with installation guides to facilitate installation on different machines. The model implementations are adopted to different machines by specifying a set of machine dependent parameters for block sizes, intersection points for alternative code sections, and parameters for important characteristics of the cache memory. They are primarily intended for a single processor on machines with local or global caches and for micro processors with on–chip caches, but they can also be parallelized by using a parallelizing compiler, or by using parallel underlying BLAS kernels.

We encourage people who perform benchmarking of a level 3 BLAS implementation to send the benchmark results to the authors (e–mail address: pol@cs.umu.se).

References

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenny, S. Ostrouchov, and D. Sorensen. *LAPACK Users Guide*. SIAM Publications, 1992. ISBN 0–89871–294–7.

[2] Intel Corporation. Paragon Basic Math Library Performance Report. Technical Report 312936-001, Intel Supercomputer Division, Beaverton, Oregon, October 1993.

[3] J. Dongarra and E. Grosse. Distribution of mathematical software by electronic mail. *Communications of the ACM*, 30(5):403–407, 1987.

[4] J. J. Dongarra, J. DuCroz, I. Duff, and S. Hammarling. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw.*, 16(1):1–17, March 1990.

[5] J. J. Dongarra, J. DuCroz, I. Duff, and S. Hammarling. Algorithm 679: A Set of Level 3 Basic Linear Algebra Subprograms: Model Implementation and Test Programs. *ACM Trans. Math. Softw.*, 16(1):18–28, March 1990.

[6] H. Grasemann. Optimization of Level 3 BLAS for SIEMENS VP Systems. Tech. rep. No. 38.89, University of Karlsruhe, Computer Center, September 1989.

[7] IBM. *Engineering and Scientific Subroutine Library, Guide and Reference*, January 1994. SC23–0526–01.

[8] B. Kågström, P. Ling, and C. Van Loan. High Performance GEMM–Based Level–3 BLAS: Sample Routines for Double Precision Real Data. In M. Durand and F. El Dabaghi, editors, *High Performance Computing II*, pages 269–281, Amsterdam, 1991. North–Holland.

[9] B. Kågström, P. Ling, and C. Van Loan. Portable High Performance GEMM–Based Level 3 BLAS. In Richard F. et al Sincovec, editor, *Parallel Processing for Scientific Computing*, pages 339–346, Philadelphia, 1993. SIAM Publications.

[10] B. Kågström, P. Ling, and C. Van Loan. GEMM–Based Level–3 BLAS: Algorithms for the Model Implementations. Report UMINF-94.13, Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden, December 1994.

[11] B. Kågström, P. Ling, and C. Van Loan. ALGORITHM XYZ. GEMM–Based Level–3 BLAS: A Portable, High–Performance Model Implementation and Benchmark. Report UMINF-95.19, Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden, 1995. To appear.

[12] B. Kågström, P. Ling, and C. Van Loan. GEMM–Based Level–3 BLAS: A Portable, High–Performance Model Implementation and Benchmark. Report UMINF-95.18, Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden, 1995. To appear.

[13] B. Kågström and C. Van Loan. GEMM–Based Level–3 BLAS. Technical Report CTC91TR47, Department of Computer Science, Cornell University, December 1989.

[14] P. Ling. A Set of High Performance Level–3 BLAS Structured and Tuned for the IBM 3090 VF and Implemented in Fortran 77. *The Journal of Supercomputing*, 7(3):323–355, September 1993.