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

Bo Kågström and *Per Ling*
Department of Computing Science
Umeå University
S-901 87 Umeå, Sweden
bokg@cs.umu.se, pol@cs.umu.se

Charles Van Loan
Department of Computer Science
Cornell University
Ithaca, New York 14853-7501
cv@cs.cornell.edu

This document only gives a short summary of the talk given at the Parallel BLAS Workshop, Knoxville, Nov. 12-14, 1995.

For more information (including performance results for different high-performance architectures) we refer to the following recent publications that also are available from WWW (see WWW page for this workshop).

GEMM-based Level 3 BLAS:
High-Performance Model Implementations and Performance Evaluation Benchmark
(*Bo Kågström, Per Ling and Charles Van Loan*, LAPACK Working Note #107)

GEMM-based Level 3 BLAS:
Installation, Tuning and Use of the Model Implementations and the Performance Evaluation Benchmark
(*Bo Kågström, Per Ling and Charles Van Loan*, LAPACK Working Note #108)

More references on the GEMM_based Level 3 BLAS (inlcuding the original concept and algorithms paper from 1989) are given in these papers.

Outline

- Basic Linear Algebra Subprograms (BLAS)
- GEMM-Based Level 3 BLAS Concept
- Design principles
- High Performance Model Implementations
- Performance Results
- GEMM-Based Level 3 BLAS Benchmark
- Benchmark Results
- Conclusions

Basic Linear Algebra Subprograms (BLAS)

Three levels of BLAS — #ops

- Level 1, vector-vector, $O(n)$
- Level 2, matrix-vector, $O(n^2)$
- Level 3, matrix-matrix, $O(n^3)$

De facto standard for basic linear algebra kernels.

Expected to be highly optimized for different architectures.

Utilized by higher level routines to gain performance and portability (e.g. LAPACK).

Why higher level BLAS ?

Reuse data at top of memory hirearchy

$\Leftrightarrow$

Minimize data movements

Level	Operation	Flop/mem refs	Mflops
1	$y := \alpha x + y$	2/3	11.9
2	$y := \alpha Ax + \beta y$	2	78.3
3	$C := \alpha AB + \beta C$	$n/2$	107.2
			(peak ≈ 138)

Optimized and tuned level 3 BLAS

$\Rightarrow$ Near to practical peak performance on hierarchical memory machines

$\Rightarrow$ Limit machine dependencies to few computational kernels

Level 3 BLAS

General matrix multiply
(_GEMM)

$C \leftarrow \alpha AB + \beta C$
$C \leftarrow \alpha A^T B + \beta C$
$C \leftarrow \alpha AB^T + \beta C$
$C \leftarrow \alpha A^T B^T + \beta C$

Symmetric rank updates
(_SYRK and _SYR2K)

$C \leftarrow \alpha A^T A + \beta C$
$C \leftarrow \alpha AA^T + \beta C$
$C \leftarrow \alpha AB^T + \alpha BA^T + \beta C$
$C \leftarrow \alpha A^T B + \alpha B^T A + \beta C$

Triangular multiply
(_TRMM)

$C \leftarrow \alpha AC$
$C \leftarrow \alpha A^T B$
$C \leftarrow \alpha CA$
$C \leftarrow \alpha CA^T$

Triangular Solve
(_TRSM)

$C \leftarrow \alpha A^{-1} C$
$C \leftarrow \alpha A^{-T} C$
$C \leftarrow \alpha CA^{-1}$
$C \leftarrow \alpha CA^{-T}$

Symmetric matrix multiply
(_SYMM)

$C \leftarrow \alpha AB + \beta C$
$C \leftarrow \alpha BA + \beta C$

GEMM-Based Level 3 BLAS Concept

Level 3 BLAS reorganized to be rich in general matrix multiply and add (GEMM) operations:

$$C \leftarrow \alpha \ \mathrm{op}(A) \ \mathrm{op}(B) + \beta C,$$

$$\mathrm{op}(X) = X \ \text{ or } X^T$$

α and β are scalars

Remaining arithmetic operations performed by some level 1 and level 2 BLAS operations

$\Rightarrow$ Only one highly optimized level 3 BLAS routine, _GEMM, required

$\Rightarrow$ Portability between different architectures

$\Rightarrow$ Limited effort for improved and new architectures

Design principles

Small number of underlying routines

- Level 3: _GEMM
- Level 2: _GEMV, _SYR, _TRMV, _TRSV
- Level 1: _COPY, _SCAL, _AXPY

Alternative code sections
(that perform the same task)

- To fully utilize the performance characteristics of the underlying routines.

- The choice is between _GEMV and another Level 2 routine (typically for diagonal blocks)

Blocking strategy

- Offset block positioning

- Update policy

Strive to enhance *locality of reference*

Sequence of operations references are kept local to limited portions of consecutive addresses in memory.

Local arrays for blocks of matrices

- To enable efficient cache reuse. Local arrays provide contiguous storage to eliminate problems with alignment and critical leading dimensions of arrays.

- To facilitate stride one computations by transposing blocks of matrices in local arrays.

- To enhance the use of _GEMM by changing storage format for blocks of matrices, e.g. *symmetric* $\rightarrow$ *general* storage format.

Level 3 performance with Level 2 BLAS

- Through carefully sized blocks kept in cache between multiple calls to Level 2 routines (mainly _GEMV).

Machine-specific parameters

- Internal block sizes.
- Intersection points for alternative codes.
- Cache characteristics.

Values assigned at compile time.
Provide guidelines and program that automatically change parameter values.
Benchmark used for tuning of parameters.

Auxiliary Routines

Besides LSAME and XERBLA two additional routines:

- _BIGP

Determines which of two alternative code sections that will be fastest for a particular problem configuration (intersection points user-specified).

- _CLD

Determines whether the size of the leading dimension of a 2D array is appropriate for the target memory hierarchy (based on cache characteristics, user-specified).

GEMM-Based Level 3 BLAS
High Performance Model Implementations

In Fortran 77 (real, double precision, complex, and double complex).

Designed to handle a memory hierarchy efficiently.

Processor and compiler sensitive code concentrated to underlying BLAS.

Designed for high performance and portability between different HPC platforms.
Easy to install.

GEMM-Based Level 3 model implementations offer:

- efficient use of arithmetic instructions through underlying routines,
- register reuse through efficient use of load and store instructions in underlying routines,
- efficient cache reuse, through internal blocking, use of local arrays, and through _GEMM.
- stride one computations where a large stride could cause severe performance degradation,
- parallelism, through automatic parallelization by a compiler, or by using parallel underlying BLAS kernels,
- opportunity for unconventional underlying algorithms like, Strassens, Winograds, or 3M based complex algorithms.

Installation

To facilitate installation, the source code comes with

- an installation guide
- a program to facilitate quick changes of the machine-specific parameters
- a benchmark program for performance evaluation and comparison with other implementations

Source code available from netlib, or by anonymous ftp from Umeå.

GEMM-Based Level 3 BLAS Benchmark

Measures performance and compares with GEMM-based level 3 BLAS. Output shows

- a collected mean value statistic of the performance of the user supplied level 3 BLAS
- detailed performance of GEMM-based level 3 BLAS
- detailed performance of user supplied level 3 BLAS library
- detailed performance of _GEMM
- GEMM-efficiency (user BLAS/_GEMM)
- GEMM-ratio (GEMM-based/user BLAS)

Conclusions

- GEMM-based concept and algorithms are very competitive for implementing a portable high performance level 3 BLAS library

- Portable to a wide range of high performance architecture's, e.g., RISC and Vector machines with memory hierarchies.

- Portability managed via (a few) architecture dependent parameters which determines partitioning in _GEMM and Level 2 operations

- Performance results show the intricacies in hand-tuning a more complicated Level 3 kernel (e.g. DTRSM on IBM 3090 VF and DSYR2K on IBM RS/6000)

- GEMM-based approach is the correct level of abstraction
  - Portable
  - Near to practical peak performance