IMPLEMENTING BLAS LEVEL 3 ON THE CAP–II

PETER E. STRAZDINS AND RICHARD P. BRENT

Abstract

The Basic Linear Algebra Subprogram (BLAS) library is widely used in many supercomputing applications, and is used to implement more extensive linear algebra subroutine libraries, such as LINPACK and LAPACK. The use of BLAS aids in the clarity, portability and maintenance of mathematical software. BLAS level 1 routines involve vector-vector operations, level 2 routines involve matrix-vector operations, and level 3 routines involve matrix-matrix operations. To take advantage of the high degree of parallelism of architectures such as the CAP-II, BLAS level 3 routines are desirable. These routines are not I/O bound; for $n \times n$ matrices, the order of arithmetic operations is $O(n^3)$ whereas the order of I/O operations is only $O(n^2)$.

We are concerned with implementing BLAS level 3 for real matrices on the CAP-II, with emphasis on obtaining the highest possible performance, without sacrificing numerical stability. While the CAP-II has many features that make it very well-suited for this purpose, there are also many new challenges in implementing BLAS level-3 on a distributed memory parallel computer (these are currently being considered also by the authors of BLAS-3, who designed it primarily for cache and shared memory architectures).

One such challenge is its external interface: BLAS-3 subroutines can be called by the host program, with the CAP array used like a (very powerful) floating point unit. Alternatively, (CAP cell equivalents of) BLAS-3 subroutines may be called by the cell programs; this approach is more efficient but deviates from the BLAS standards. These issues are discussed.

We also discuss the high-level design of basic parallel matrix multiplication, transposition and triangular matrix inversion algorithms to be used by the BLAS-3 subroutines. With the efficient row/column broadcast available on the CAP-II using wormhole routing, "semi-systolic" algorithms, with a low startup time, appear to be superior to other algorithms. It is hoped that input from the workshop can help to finalize details of the high-level design.

On the lower levels of design, optimization of cell program codes (e.g. optimizing inner product or gaxpy loops, use of the SPARC cache) must be considered. Also relevant is the degree of optimization possible from the available CAP-II cell program compilers.

Comments

Only the Abstract is given here. The full paper appeared as [4]. For related, later work, see [3, 5].

References

[1] R. P. Brent (editor), *Proceedings of the Second Fujitsu-ANU CAP Workshop*, Department of Computer Science, Australian National University, November 1991, 254 pp. rpb129.

1991 *Mathematics Subject Classification.* Primary 65Y05; Secondary 65F05, 65Y10.

Key words and phrases. BLAS, basic linear algebra subroutines, parallel computation, distributed memory, local memory, MIMD, Fujitsu AP1000, Gaussian elimination, partial pivoting, matrix multiplication.

[2] R. P. Brent and M. Ishii (editors), *Proceedings of the First Fujitsu-ANU CAP Workshop*, Fujitsu Research Laboratories, Kawasaki, Japan, November 1990, 205 pp. rpb123.

[3] R. P. Brent and P. E. Strazdins, "Implementation of the BLAS level 3 and Linpack benchmark on the AP 1000", *Fujitsu Scientific and Technical Journal* 29, 1 (March 1993), 61–70. rpb136.

[4] P. E. Strazdins and R. P. Brent, "Implementing BLAS level 3 on the CAP-II", in [2, paper 12, pp. 1–9]. rpb121.

[5] P. E. Strazdins and R. P. Brent, "The implementation of BLAS level 3 on the AP 1000: Preliminary report", in [1, H1–H17]. rpb131.

Department of Computer Science and Computer Sciences Laboratory, Australian National University, Canberra, ACT 0200

E-mail address: {peter,rpb}@cs.anu.edu.au