Exposing Inner Kernels and Packed Storage for Fast Dense Linear Algebra Codes*

José R. Herrero**

Computer Architecture Department
Universitat Politècnica de Catalunya
Barcelona, Spain
josepr@ac.upc.edu

Abstract. Efficient execution on processors with multiple cores requires the exploitation of parallelism within the processor. For many dense linear algebra codes this, in turn, requires the efficient execution of codes which operate on relatively small matrices. Efficient implementations of dense linear algebra codes exist (BLAS libraries). They rearrange (pack) data internally so that data can be streamed from the level 2 cache and often get excellent performance when used with medium and large matrices. However, calls to BLAS libraries introduce large overheads when they operate on small matrices. High performance implementations of dense linear algebra codes can be achieved by replacing calls to standard BLAS libraries with calls to specialized inner kernels which work on small data submatrices which are already packed in the proper way.

Key words: Inner kernels, packed storage, register blocks.

1 Introduction

1.1 Packed Storage for Register Blocking

In [1], Gustavson states the need to reorder square blocks so that data loaded into the level 1 cache can enter the register file in an optimal way. Such approach was followed in the creation of a high performance implementation of matrix multiplication on the IBM Power PC 440 [2]. A similar approach is also used in Goto's library [3] for a variety of platforms.

1.2 Specialized Inner Kernels

The specialization of the inner kernels avoids performing unnecessary operations repetitively. In addition, it simplifies the code allowing for more opportunities for automatic optimization. Working with simple square blocks it is possible to produce efficient inner kernels with the help of an optimizing compiler.

* This work was supported by the Ministerio de Educación y Ciencia of Spain (TIN2007-60625).

** Currently on sabbatical leave at Barcelona Supercomputing Center

When these kernels are called directly from linear algebra codes which store matrices using non-canonical data structures the overhead is very low. This happens because there are no costs associated to copying data and checking certain parameters.

In [4], Herrero shows that the performance obtained from the resulting Cholesky factorization approaches that of a hand-optimized implementation in which most representative parts of the code are written in assembly code and data is packed for efficient use of the register file (Goto BLAS).

1.3 Non-Canonical Storage for Multi-core Architectures

Recent work on the parallelization of dense linear algebra codes on multi-core architectures uses new data structures for matrices. Basically, matrices are stored as a set of submatrices which are kept as square blocks [5–7]. Such codes either call BLAS routines or vanilla kernels and could benefit from using specialized inner kernels with packed storage.

2 Specialized Kernels and Register Blocking

In this talk we will present ongoing work on the modification of the Small Matrix Library (SML) [8, 9] developed previously by the author. New routines operate on small matrices stored in a way that allows for register blocking and facilitates data streaming. Resulting codes can avoid most of the overhead and have high performance.

References

1. Gustavson, F.G.: Algorithm Compiler Architecture Interaction Relative to Dense Linear Algebra. Technical Report RC23715 (W0509-039), IBM, T.J. Watson (September 2005)
2. Chatterjee, S., Bachega, L.R., Bergner, P., Dockser, K.A., Gunnels, J.A., Gupta, M., Gustavson, F.G., Lapkowski, C.A., Liu, G.K., Mendell, M., Nair, R., Wait, C.D., Ward, T.J.C., Wu, P.: Design and exploitation of a high-performance SIMD floating-point unit for Blue Gene/L. IBM Journal of Research and Development **49**(2/3) (March/May 2005) 377–391
3. Goto, K., van de Geijn, R.A.: Anatomy of a high-performance matrix multiplication. ACM Transactions on Mathematical Software **34**(3) (September 2007)
4. Herrero, J.R.: New data structures for matrices and specialized inner kernels: Low overhead for high performance. In: Int. Conf. on Parallel Processing and Applied Mathematics. (PPAM'07). LNCS (To appear). Lecture Notes in Computer Science, Springer-Verlag (September 2007)
5. Chan, E., Zee, F.V., van de Geijn, R., Quintana-Ortí, E.S., Quintana-Ortí, G.: Satisfying your dependencies with SuperMatrix. In: IEEE Cluster 2007. (2007) 92–99
6. Buttari, A., Langou, J., Kurzak, J., Dongarra, J.: A class of parallel tiled linear algebra algorithms for multicore architectures. CoRR **abs/0709.1272** (2007) informal publication.

7. Perez, J.M., Badia, R.M., Labarta, J.: A flexible and portable programming model for SMP and multi-cores. Technical report, Barcelona Supercomputing Center - Centro Nacional de Supercomputación (June 2007) Technical report 03/2007.
8. Herrero, J.R., Navarro, J.J.: Automatic benchmarking and optimization of codes: an experience with numerical kernels. In: Int. Conf. on Software Engineering Research and Practice, CSREA Press (June 2003) 701–706
9. Herrero, J.R., Navarro, J.J.: Compiler-optimized kernels: An efficient alternative to hand-coded inner kernels. In: Proceedings of the International Conference on Computational Science and its Applications (ICCSA). LNCS 3984. (May 2006) 762–771