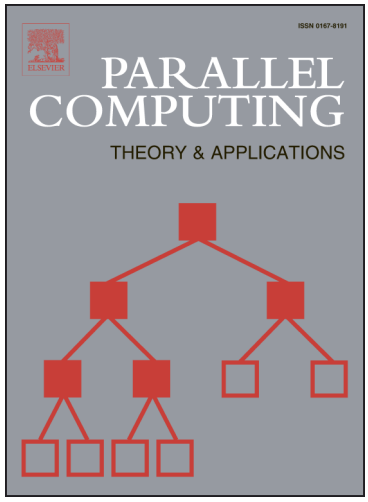

Efficient Multithreaded Untransposed, Transposed or Symmetric Sparse Matrix-Vector Multiplication with the Recursive Sparse Blocks Format

Michele Martone

PII: S0167-8191(14)00038-6
DOI: http://dx.doi.org/10.1016/j.parco.2014.03.008
Reference: PARCO 2171

To appear in: *Parallel Computing*

Please cite this article as: M. Martone, Efficient Multithreaded Untransposed, Transposed or Symmetric Sparse Matrix-Vector Multiplication with the Recursive Sparse Blocks Format, *Parallel Computing* (2014), doi: http://dx.doi.org/10.1016/j.parco.2014.03.008

Efficient Multithreaded Untransposed, Transposed or Symmetric Sparse Matrix-Vector Multiplication with the Recursive Sparse Blocks Format

Michele Martone[a,1,*]

[a] *Max Planck Society Institute for Plasma Physics, Boltzmannstrasse 2, D-85748 Garching bei München, Germany*

Abstract

In earlier work we have introduced the *"Recursive Sparse Blocks"* (RSB) sparse matrix storage scheme oriented towards cache efficient matrix-vector multiplication (*SpMV*) and triangular solution (*SpSV*) on cache based shared memory parallel computers. Both the transposed (*SpMV_T*) and symmetric (*SymSpMV*) matrix-vector multiply variants are supported. RSB stands for a meta-format: it recursively partitions a rectangular sparse matrix in quadrants; leaf submatrices are stored in an appropriate traditional format — either *Compressed Sparse Rows* (CSR) or *Coordinate* (COO). In this work, we compare the performance of our RSB implementation of *SpMV*, *SpMV_T*, *SymSpMV* to that of the state-of-the-art Intel Math Kernel Library (MKL) CSR implementation on the recent Intel's Sandy Bridge processor. Our results with a few dozens of real world large matrices suggest the efficiency of the approach: in all of the cases, RSB's *SymSpMV* (and in most cases, *SpMV_T* as well) took less than half of MKL CSR's time; *SpMV*'s advantage was smaller. Furthermore, RSB's *SpMV_T* is more scalable than MKL's CSR, in that it performs almost as well as *SpMV*. Additionally, we include comparisons to the state-of-the art format Compressed Sparse Blocks (CSB) implementation. We observed RSB to be slightly superior to CSB in *SpMV_T*, slightly inferior in *SpMV*, and better (in most cases by a factor of two or more) in *SymSpMV*. Although RSB is a non-traditional storage format and thus needs a special constructor, it can be assembled from CSR or any other similar row-ordered representation arrays in the time of a few dozens of matrix-vector multiply executions. Thanks to its significant advantage over MKL's CSR routines for symmetric or transposed matrix-vector multiplication, in most of the observed cases the assembly cost has been observed to amortize with fewer than fifty iterations.

Keywords: sparse matrix-vector multiply, symmetric matrix-vector multiply, transpose matrix-vector multiply, shared memory parallel, cache blocking, sparse matrix assembly

1. Introduction and Related Literature

Many scientific and engineering problems require the solution of large *sparse* linear systems of equations; that is, systems where the number of equations largely outnumbers the average number of unknowns per equation. Since most of the entries in the (floating point number) coefficient matrices associated to such systems are zeroes, it is advantageous to represent them in a computer by their non-zero coefficients only. Such matrices (and their data structures in a computer) are therefore called *sparse*. A class of techniques for solving such systems is that of *iterative methods* [1]. Their computational core is largely based on repeated Sparse Matrix-Vector Multiplication (*SpMV*, defined as "$y \leftarrow y + A\,x$"; with A being the matrix, and x, y vectors) executions. Methods as BiCG or QMR (see Barrett et al. [2, Ch. 2]) or *Krylov balancing* algorithms (see Bai et al. [3, Alg. 7.1]) require computation of the *transpose product* as well (*SpMV_T*, defined as "$y \leftarrow y + A^T\,x$"). Availability of an efficient algorithm for *SpMV_T* eliminates the need for an explicit transposed matrix representation. Many applications give rise to symmetric matrices (that

*Corresponding Author
Email address: michele.martone@ipp.mpg.de (Michele Martone)