LIBRSB: Multicore Sparse Matrix Performance across Languages and Architectures

Michele MARTONE
michele.martone@lrz.de
Leibniz Supercomputing Centre, Garching bei München, Germany

LIBRSB: Universal Sparse BLAS Library

https://librsb.sf.net

A portable Sparse BLAS

"Sparse Basic Linear Algebra Subroutines"

▌for multicore CPUs (node-level)
▌one library, multiple APIs
▌this post: LIBRSB overview and ongoing work

Numerical Techniques of Interest
• iterative methods: block Krylov
• require efficient Sparse Matrix-Matrix multiplication aka SpMM

SpMM in matrix form (m aka NRHS aka number of right hand sides):

```
C = A · B
```

LIBRSB

▌project page: http://librsb.sf.net
• >100KLOC of C99, OpenMP, and modern templated C++
• node-level shared-memory-parallel operations, for:
  – Sparse BLAS: matrix assembly/destroy, SpMM, triangular solve
  – interactive applications (update, sparse-sparse ops, conversions)
  – distributed-memory applications (block extract, update, etc.)
▌with own interface: C/C++ and Fortran
▌with Sparse BLAS interface (BLAS Technical Forum Standard)
▌with interfaces for GNU OCTAVE and PYTHON interpreters
▌GNU GPLv3-licensed
▌Free software, available via SPACK, CYCLES

Modern C++ interface (rbz.hpp)

```
#include <rbz.hpp>
#include <stdlib>
#include <iostream>
#include <vector>
#include <Windows.h>

RsbMatrix A;
RsbMatrix B;
RsbMatrix C;

int main()
{
    // ditto
    // double of RsbMatrix
    const RsbLib liboctave;
    RsbMatrix A;
    // a
    // usage is styled after the
    // (sparse)
    // and others use
    // (int)
    // boxes are
    // average
    include
    // mtx.
    // int
    // //
    // top
    // redder
    include
    // Z-ordered
    // above
    RSB_FLAG_WANT_ROW_MAJOR_ORDER
    int
    main() {
        // double
        of
        RsbMatrix
        const
        RsbLib
        and
        for
        Notice
        using namespace ::rsb;
    }
```

Recursive Sparse Blocks (RSB) Layout

• for large matrices (uses cache locality, coarse thread parallelism)
• supports autotuning (layout adjusted to maximize performance)

Figure 1: Instance of classical test matrix layernz2 (144 × 144, 664500 nonzeros). Bank-低位化rines are sparse blocks, and are Z-ordered. Columns have fewer than maxrow. Blocks have more. Either in "Co-ordinate" format (COO) or "Compressed Sparse Rows" (CSR). Blocks rows (LHS) and columns (RHS) ranges evidenced (left and top side).

Figure 3: The sparse usage is styled after the sparse built-in. So most of operators (×××,(...)...) work the same way.

GNU OCTAVE + iliboctave + LIBRSB = SparseRSB

▌ GNU OCTAVE
▌ MATLAB like interactive numerical language
▌ iliboctave: access OCTAVE via C++

```
\texttt{\textbackslash{}octave} -d \\texttt{--sparse} \\
\texttt{\textbackslash{}sparse}\ (\texttt{\textbackslash{}read}\ \texttt{\textbackslash{}write})
```

Python + Cython + LIBRSB = PyRSB

▌ SciPy
▌ popular PYTHON scientific computing API
▌ Cython
▌ optimizing static compiler for C extensions to PYTHON

```
import scipy
import numpy
```

New: ongoing work

▌ Improved performance of SpMM CSR kernels:
  – AVX512 intrinsics (doy-rows, unsymmetric, untransposed)
  – more specializations
  – autotuning challenges: e.g. use CSR if more performant than RSB

Remarks

Two RSB block layouts (a corner left, a four right) of a Machine Learning matrix sized 574 × 238 and 48M nonzero (courtesy Dr. Diego De Cao). The matrix is particularly favorable to RSB over CSR. Determining optimal blocking and thread sizes is aided by the autotuning functionality.

Acknowledgements

▌ This work has received funding by PRACE IP6/MPI9 and PRACE High Level Support Teams.
▌ Development and test of LIBRSB utilizes systems of the BEAST (Bavarian Energy Architecture & Software Testbed) at LRZ.