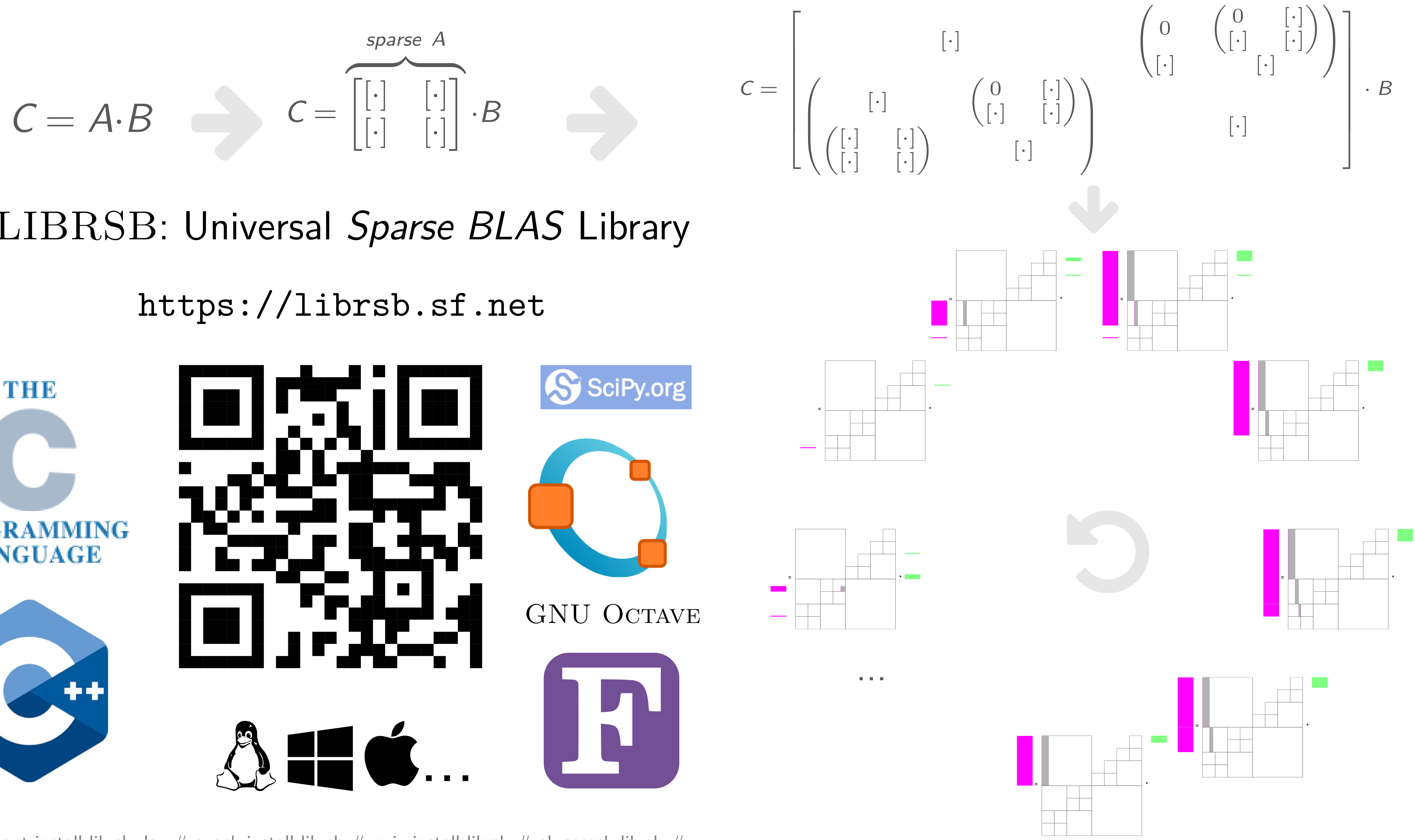
 https://doi.org/10.5281/zenodo.8121192

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

apt install librsb-dev # spack install librsb # guix install librsb # eb search librsb

Programming languages-related images with permission from www.octave.org, scipy.org, cplusplus.org, fortran-lang.org, wikipedia.org

A portable Sparse BLAS

- "Sparse *Basic Linear Algebra Subroutines*"
- for multicore CPUs (node-level)
- one library, multiple APIs
- **this poster: 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*):

$$\underbrace{\begin{bmatrix} c_{11} & \dots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \dots & c_{nm} \end{bmatrix}}_{\text{updated dense } C} \leftarrow \beta \underbrace{\begin{bmatrix} c_{11} & \dots & c_{1m} \\ \vdots & \ddots & \vdots \\ c_{n1} & \dots & c_{nm} \end{bmatrix}}_{\text{dense } C} + \alpha \underbrace{\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix}}_{\text{sparse } A} \underbrace{\begin{bmatrix} b_{11} & \dots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{n1} & \dots & b_{nm} \end{bmatrix}}_{\text{dense } B}$$

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 in C/C++ and Fortran
  - with Sparse BLAS interface (BLAS Technical Forum Standard)
  - with interfaces for GNU Octave and Python interpreters
- **LGPLv3-licensed** *free software*, available via Spack, Guix-HPC, EasyBuild, and on Debian, Ubuntu, OpenSUSE, Windows,...

Recursive Sparse Blocks (RSB) Layout

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

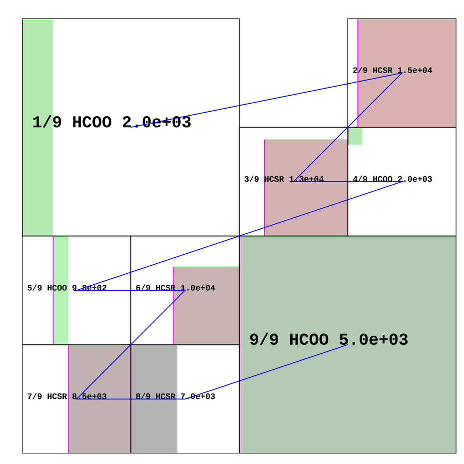


Figure 1: Instance of classical test matrix *bayer02* ($14k \times 14k, 64k$ nonzeroes). Black-bordered boxes are *sparse blocks*, and are Z-ordered. Greener have fewer nnz than average, redder have more. Either in *"Coordinate"* format (COO) or *"Compressed Sparse Rows"* (CSR). Blocks rows (LHS) and columns (RHS) ranges evidenced (left and top side).

Modern C++ interface (rsb.hpp)

```cpp
#include <rsb.hpp>
#include <vector>
#include <array>
using namespace ::rsb;

int main() {
  RsbLib rsblib;
  const int nnzA { 7 }, nrhs { 2 };
  const int nrA { 6 }, ncA { 6 };
  const std::vector<int>    IA {0,1,2,3,4,5,1};
  const int                 JA [] = {0,1,2,3,4,5,0};
  const std::vector<double> VA {1,1,1,1,1,1,2}, B(nrhs * ncA,1);
  std::array<double,nrhs * nrA> C;
  const double alpha {2}, beta {1};

  RsbMatrix<double> mtx(IA,JA,VA,nnzA);// Notice above declarations of IA,JA,VA

  mtx.spmm(RSB_TRANSPOSITION_N, alpha, nrhs, RSB_FLAG_WANT_ROW_MAJOR_ORDER, B,
    beta, C); // ditto for B and C
}
```

Figure 2: RsbMatrix's spmm() and others use C++20's std::span. Gives freedom in vectors' type choice. Wraps rsb.h.

GNU Octave + liboctave + LIBRSB = SparseRSB

- GNU Octave: a MATLAB-like interactive numerical language
- liboctave: access Octave via C++

```
octave:1> A = sparsersb ( sparse (rand (3) > .6) )
A =

Recursive Sparse Blocks  (rows = 3, cols = 3, nnz = 2 [22%])

  (3, 1) -> 1
  (3, 3) -> 1
```

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

Python + Cython + LIBRSB = PyRSB

- SciPy: popular Python scientific computing API
- Cython: *optimising static compiler* for C extensions to Python

```python
import numpy
import scipy
from rsb            import rsb_matrix
V=[ 11.,12.,22.]
I=[   0,   0,   1]
J=[   0,   1,   1]
a = rsb_matrix((V, (I, J)), [3,3])
 ...
y = y + a * x;
```

Figure 4: API styled after SciPy's widely known csr_matrix. Additionally, it offers RSB-specific functionality, e.g. blocks autotuning.

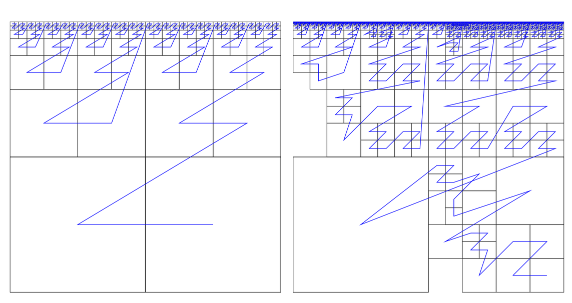
New: ongoing work

Improve performance of SpMM CSR kernels:

- AVX512 intrinsics (*by-rows*, unsymmetric, untransposed)
- more specializations
- autotuning challenges: e.g. use CSR if more performant than RSB

Remarks

Two RSB block layouts (a *coarser* left, a *finer* right) of a Machine Learning matrix sized $57k \times 230k$ and 41M nonzeroes (courtesy Dr. Diego De Cao). This matrix is particularly favourable to RSB over CSR. Determining optimal blocking and threads is eased by the autotuning functionality.



Acknowledgements

This work has received funding by *PRACE IP6/WP8* and *PRACE High Level Support Teams*.

Development and test and of LIBRSB utilizes systems of the BEAST (*Bavarian Energy Architecture & Software Testbed*) at LRZ.