How to Write Fast Numerical Code

Spring 2011

Lecture 16

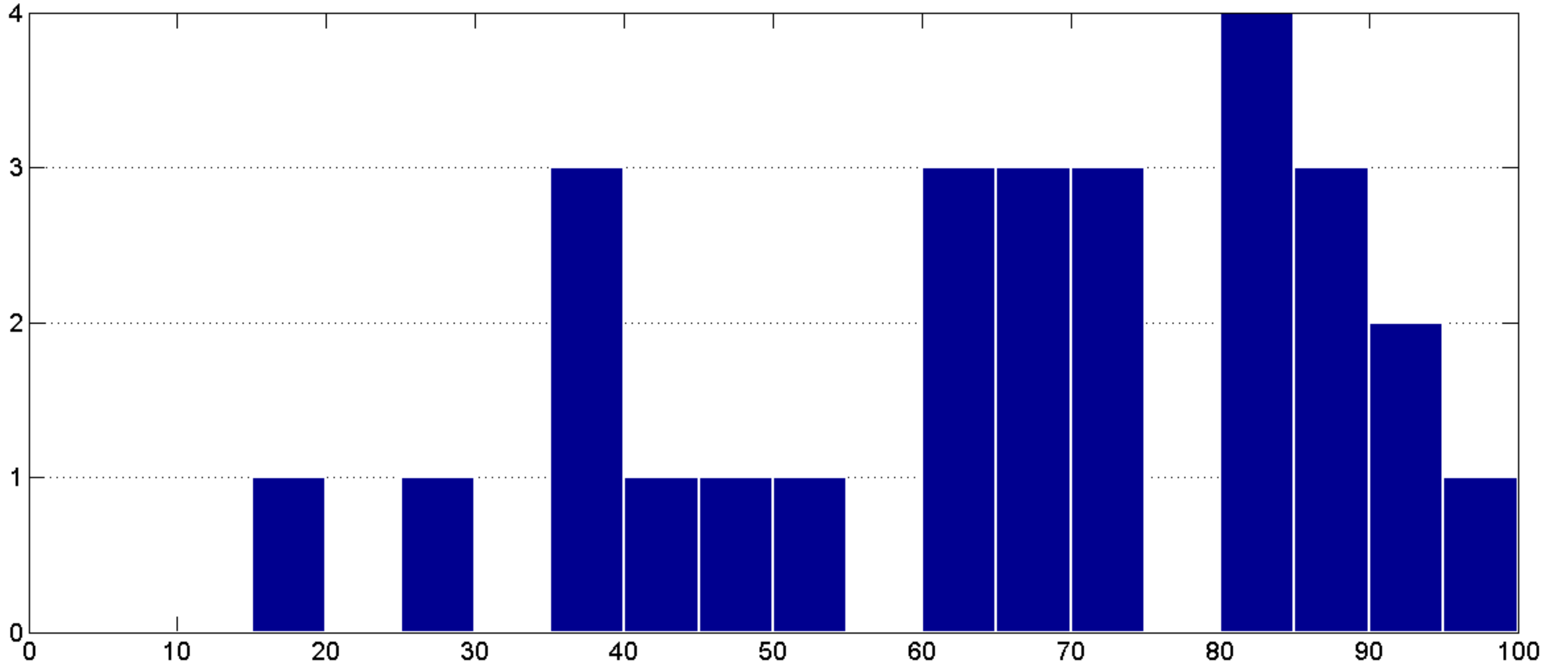
Instructor: Markus Püschel

TA: Georg Ofenbeck

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Midterm

27 people
average: 65

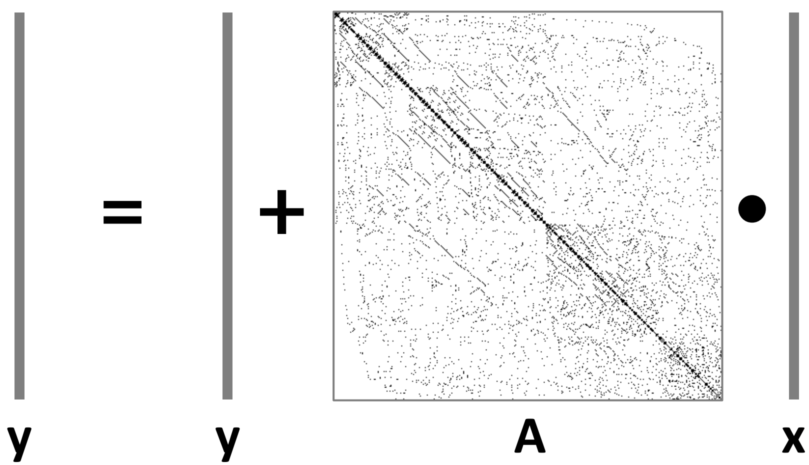


Today

- SMVM continued

Sparse MVM (SMVM)

- y = y + Ax, A sparse but known



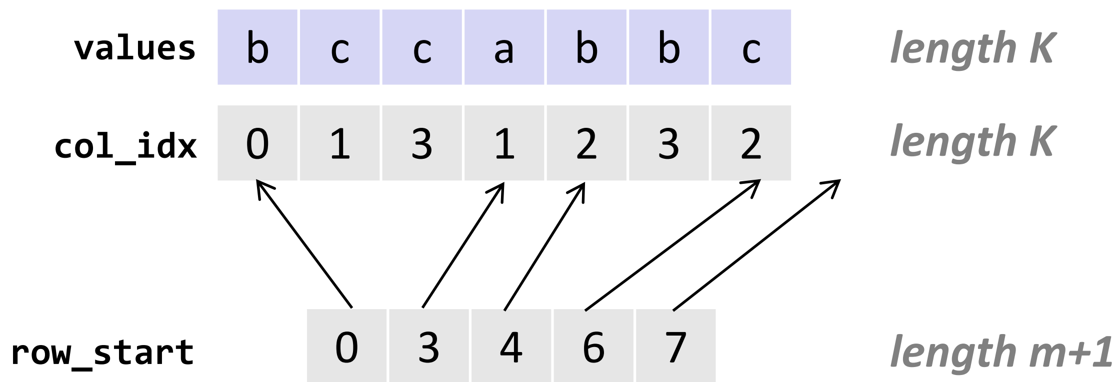
CSR

- **Assumptions:**
  - A is m x n
  - K nonzero entries

A as matrix

b	c		c
	a		
		b	b
		c	

A in CSR:

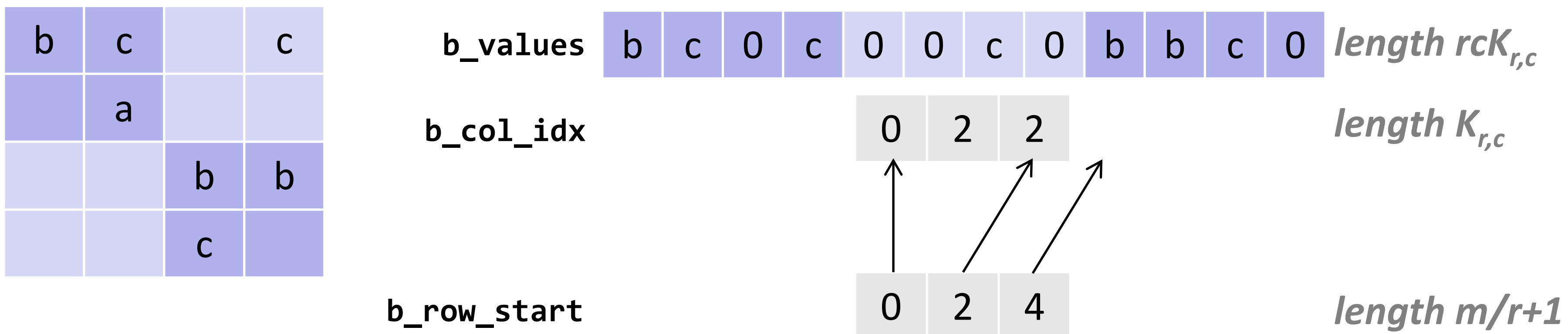


BCSR (Blocks of Size r x c)

- **Assumptions:**
  - A is m x n
  - Block size r x c
  - $K_{r,c}$ nonzero blocks

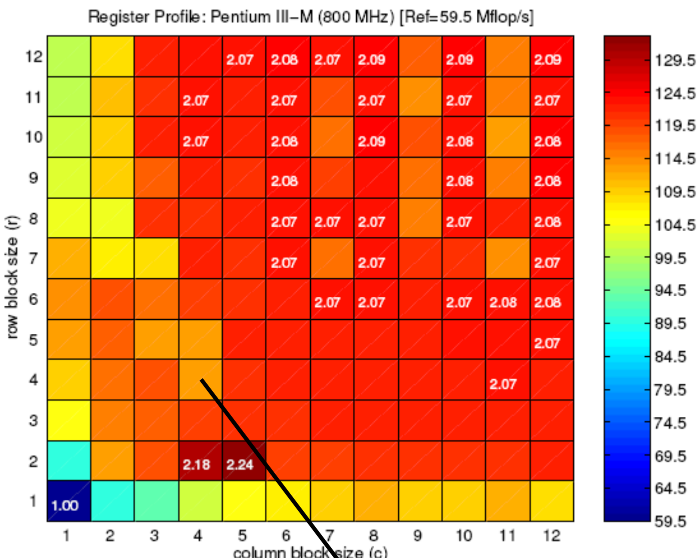
A as matrix (r = c = 2)

A in BCSR (r = c = 2):



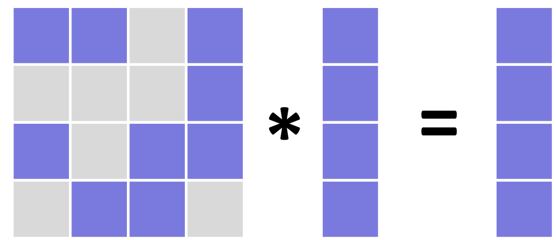
Model: Example

Gain by blocking (dense MVM)



1.4

Overhead (average) by blocking

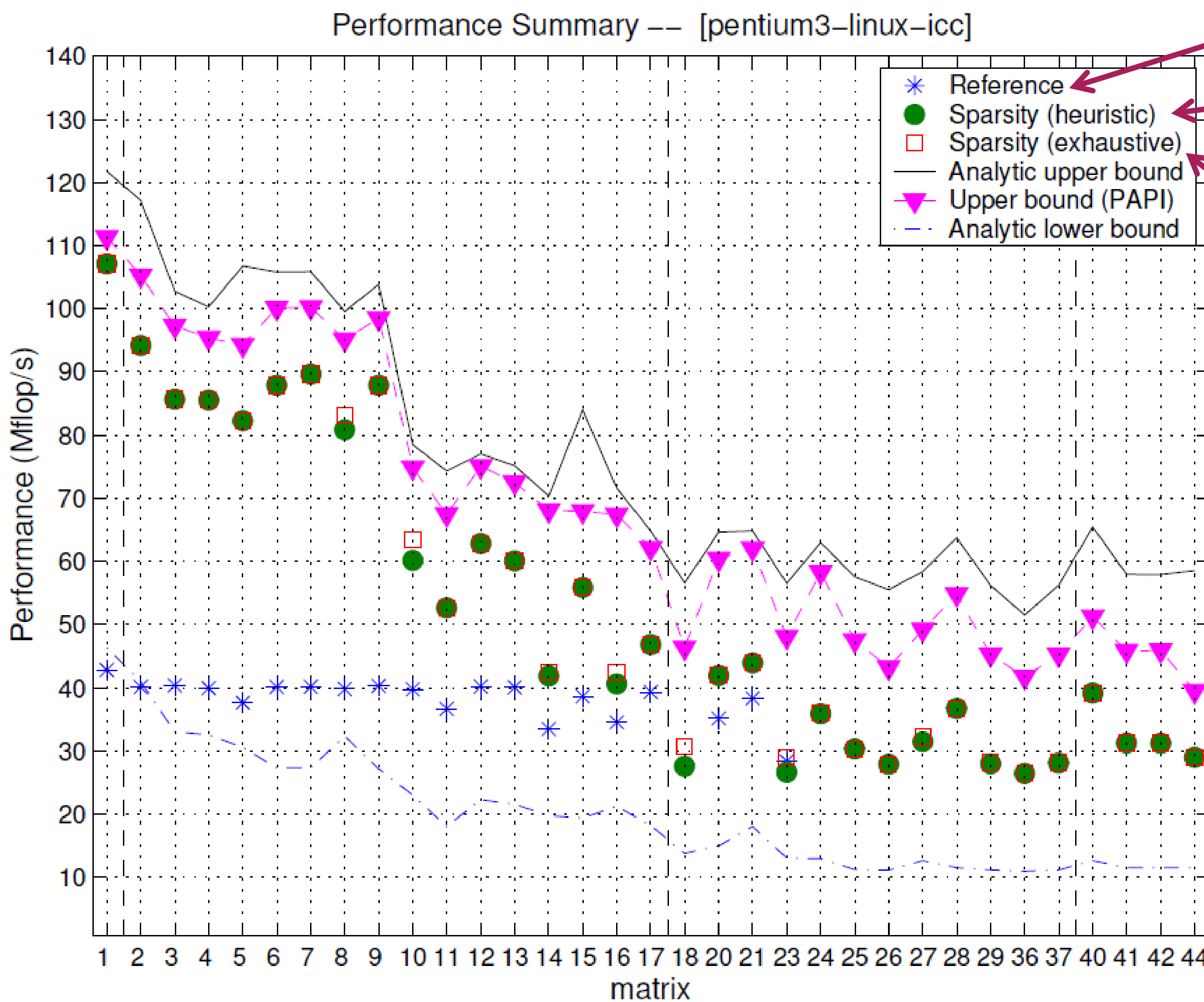


16/9 = 1.77

1.4/1.77 = 0.79 (no gain)

Model: **Doing that for all r and c and picking best**

Typical Result



CRS

BCRS model

BCSR exhaustive search

Analytical upper bound
how obtained?
(blackboard)

Figure: Eun-Jin Im, Katherine A. Yelick, Richard Vuduc. *SPARSITY: An Optimization Framework for Sparse Matrix Kernels*, *Int'l Journal of High Performance Comp. App.*, 18(1), pp. 135-158, 2004

Principles in Bebop/Sparsity Optimization

- *SMVM is memory bound*
- **Optimization for memory hierarchy = increasing locality**
  - Blocking for registers (micro-MMMs)
  - *Requires change of data structure for A*
  - Optimizations are *input dependent* (on sparse structure of A)
- **Fast basic blocks for small sizes (micro-MMM):**
  - Unrolled, scalar replacement (enables better compiler optimization)
- **Search for the fastest over a relevant set of algorithm/implementation alternatives (parameters r, c)**
  - *Use of performance model* (versus measuring runtime) to evaluate expected gain

Different from ATLAS

SMVM: Other Ideas

- Value compression
- Index compression
- Pattern-based compression
- Cache blocking
- Special scenario: Multiple inputs

Value Compression

- *Situation:* **Matrix A contains many duplicate values**
- *Idea:* **Store only unique ones plus index information**

b	c		c
	a		
		b	b
		c	

A in CSR:

`values`	b	c	c	a	b	b	c
`col_idx`	0	1	3	1	2	3	2

`row_start`	0	3	4	6	7

A in CSR-VI:

a	b	c

`values`	1	2	2	0	1	1	2
`col_idx`	0	1	3	1	2	3	2

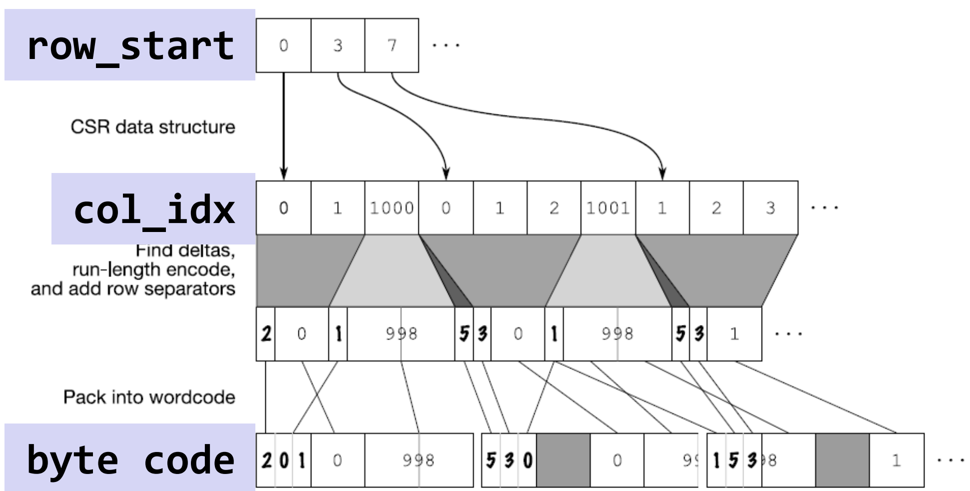
`row_start`	0	3	4	6	7

Kourtis, Goumas, and Koziris, Improving the Performance of Multithreaded Sparse Matrix-Vector Multiplication using Index and Value Compression, pp. 511-519, ICPP 2008

Index Compression

- ***Situation:*** **Matrix A contains sequences of nonzero entries**
- ***Idea:*** **Use special byte code to jointly compress `col_idx` and `row_start`**

Coding



Decoding

0: acc = acc $*$ 256 + arg;
1: col = col + acc $*$ 256 + arg; acc = 0;
emit_element(row, col); col = col + 1;
2: col = col + acc $*$ 256 + arg; acc = 0;
emit_element(row, col);
emit_element(row, col + 1); col = col + 2;
3: col = col + acc $*$ 256 + arg; acc = 0;
emit_element(row, col);
emit_element(row, col + 1);
emit_element(row, col + 2); col = col + 3;
4: col = col + acc $*$ 256 + arg; acc = 0;
emit_element(row, col);
emit_element(row, col + 1);
emit_element(row, col + 2);
emit_element(row, col + 3); col = col + 4;
5: row = row + 1; col = 0;

Willcock and Lumsdaine, Accelerating Sparse Matrix Computations via Data Compression, pp. 307-316, ICS 2006

Pattern-Based Compression

- *Situation:* After blocking A, many blocks have the same nonzero pattern
- *Idea:* Use special BCSR format to avoid storing zeros;
  needs specialized micro-MVM kernel for each pattern

A as matrix

b	c		c
	a		
		b	b
		c	

Values in 2 x 2 BCSR

b	c	0	c	0	0	c	0	b	b	c	0

Values in 2 x 2 PBR

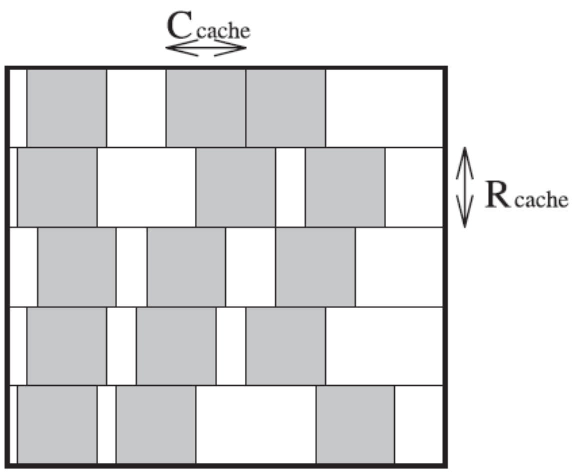
b	c	c	c	b	b	c

+ bit string: 1101 0100 1110

Belgin, Back, and Ribbens, Pattern-based Sparse Matrix Representation for Memory-Efficient SMVM Kernels, pp. 100-109, ICS 2009

Cache Blocking

- **Idea: divide sparse matrix into blocks of sparse matrices**

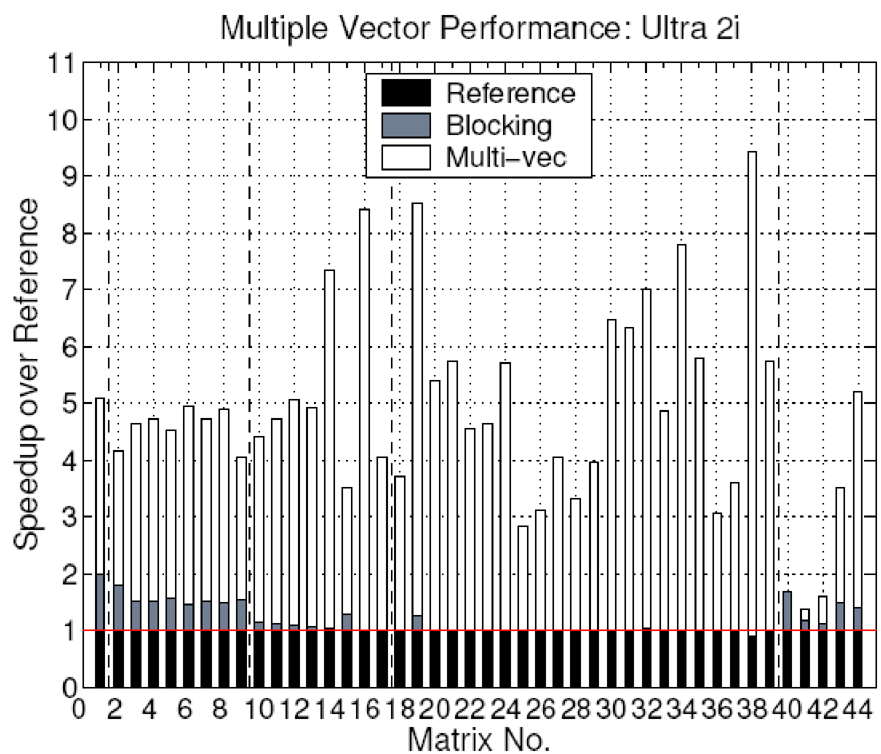


- **Experiments:**
  - Requires very large matrices (x and y do not fit into cache)
  - Speed-up up to 2.2x, only for few matrices, with 1 x 1 BCSR

Figure: *Eun-Jin Im, Katherine A. Yelick, Richard Vuduc. SPARSITY: An Optimization Framework for Sparse Matrix Kernels, Int'l Journal of High Performance Comp. App., 18(1), pp. 135-158, 2004*

Special scenario: Multiple inputs

- **Situation: Compute SMVM y = y + Ax for several independent x**
- **Blackboard**
- **Experiments:**
  up to 9x speedup for 9 vectors



Source: Eun-Jin Im, Katherine A. Yelick, Richard Vuduc. *SPARSITY: An Optimization Framework for Sparse Matrix Kernels*, *Int'l Journal of High Performance Comp. App.*, 18(1), pp. 135-158, 2004