

How to Write Fast Numerical Code

Spring 2013

Lecture: Memory bound computation, sparse linear algebra, OSKI

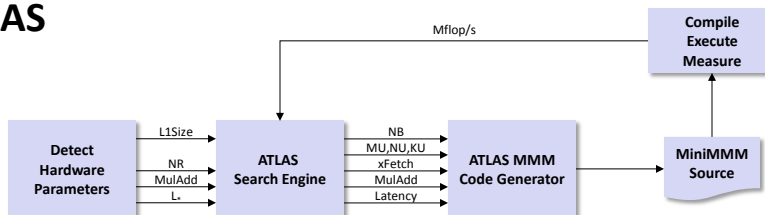
Instructor: Markus Püschel

TA: Georg Ofenbeck & Daniele Spampinato

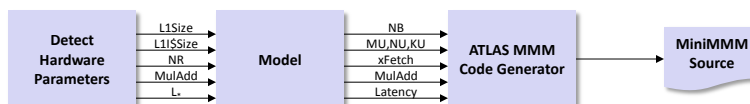
ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ATLAS



Model-Based ATLAS



source: Pingali, Yotov, Cornell³U.

Principles

- **Optimization for memory hierarchy**
 - Blocking for cache
 - Blocking for registers
- **Basic block optimizations**
 - Loop order for ILP
 - Unrolling + scalar replacement
 - Scheduling & software pipelining
- **Optimizations for virtual memory**
 - Buffering (copying spread-out data into contiguous memory)
- **Autotuning**
 - Search over parameters (ATLAS)
 - Model to estimate parameters (Model-based ATLAS)
- *All high performance MMM libraries do some of these (but possibly in a different way)*

3

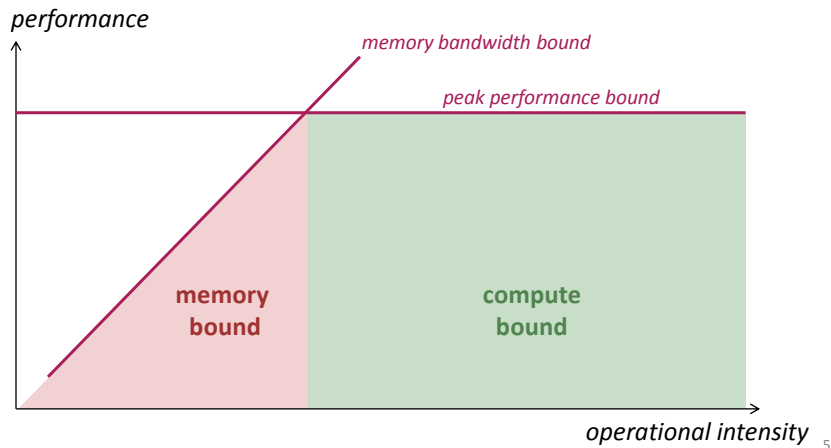
Today

- **Memory bound computations**
- **Sparse linear algebra, OSKI**

4

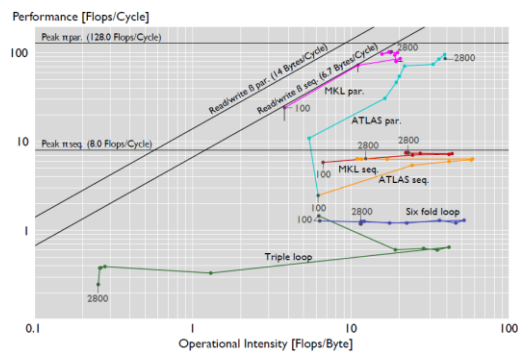
Memory Bound Computation

- Data movement, not computation, is the bottleneck
- Typically: Computations with operational intensity $I(n) = O(1)$



Memory Bound Or Not? Depends On ...

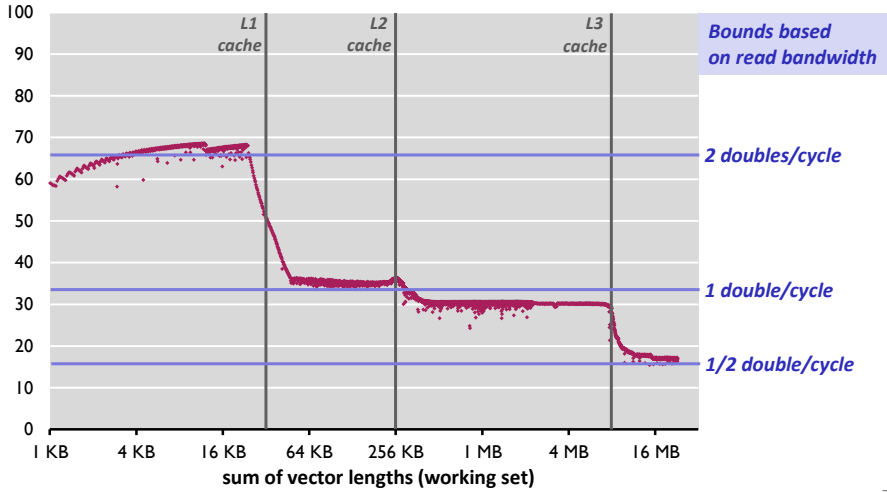
- **The computer**
 - Memory bandwidth
 - Peak performance
- **How it is implemented**
 - Good/bad locality
 - SIMD or not
- **How the measurement is done**
 - Cold or warm cache
 - In which cache data resides
 - See next slide



Example 1: BLAS 1, Warm Data & Code

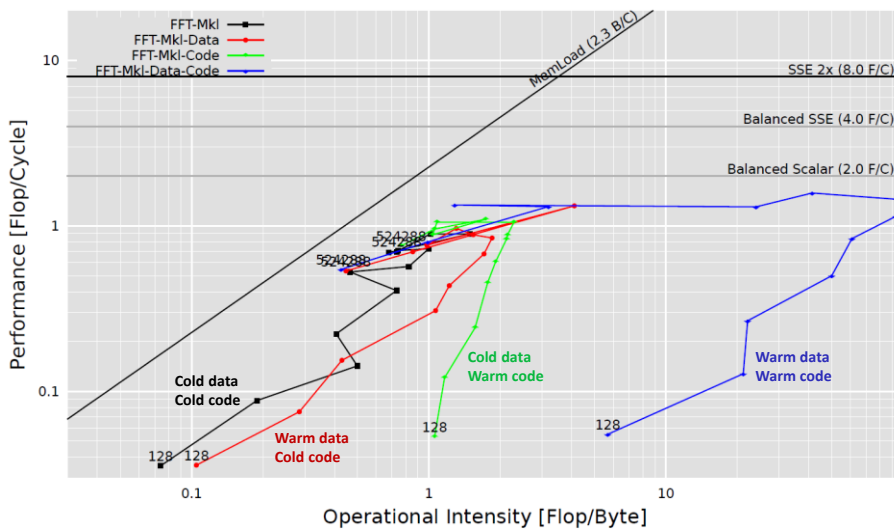
$z = x + y$ on Core i7 (Nehalem, one core, no SSE), icc 12.0 /O2 /fp:fast /Qipo

Percentage peak performance (peak = 1 add/cycle)



7

Example 3: Cold/Warm Data & Code



preliminary plots roofline goal
developed in our group by R. Steinmann

Sparse Linear Algebra

- Sparse matrix-vector multiplication (MVM)
- Sparsity/Bebop/OSKI

- **References:**

- 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
- Vuduc, R.; Demmel, J.W.; Yelick, K.A.; Kamil, S.; Nishtala, R.; Lee, B.; *Performance Optimizations and Bounds for Sparse Matrix-Vector Multiply*, pp. 26, Supercomputing, 2002
- [Sparsity/Bebop](#) website

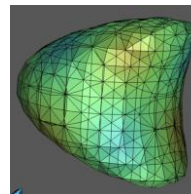
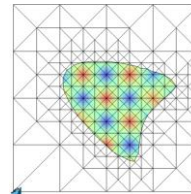
9

Sparse Linear Algebra

- Very different characteristics from dense linear algebra (LAPACK etc.)

- **Applications:**

- finite element methods
- PDE solving
- physical/chemical simulation (e.g., fluid dynamics)
- linear programming
- scheduling
- signal processing (e.g., filters)
- ...



- **Core building block: Sparse MVM**

Graphics: http://aam.mathematik.uni-freiburg.de/IAM/homepages/clqys/projects/unfitted-meshes_en.html

CSR

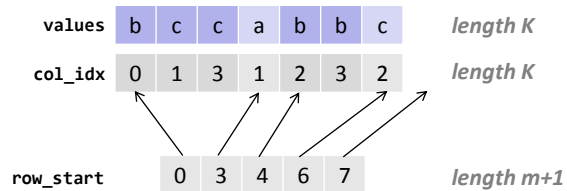
Assumptions:

- A is $m \times n$
- K nonzero entries

A as matrix

b	c		c
	a		
		b	b
		c	

A in CSR:



Storage:

- K doubles + (K+m+1) ints = $\Theta(\max(K, m))$
- Typically: $\Theta(K)$

13

Sparse MVM Using CSR

$$y = y + Ax$$

```
void smvm(int m, const double* values, const int* col_idx,
          const int* row_start, double* x, double* y)
{
    int i, j;
    double d;

    /* loop over m rows */
    for (i = 0; i < m; i++) {
        d = y[i]; /* scalar replacement since reused */

        /* loop over non-zero elements in row i */
        for (j = row_start[i]; j < row_start[i+1]; j++)
            d += values[j] * x[col_idx[j]];
        y[i] = d;
    }
}
```

CSR + sparse MVM: Advantages?

14

CSR

■ Advantages:

- Only nonzero values are stored
- All three arrays for A (**values**, **col_idx**, **row_start**) accessed consecutively in MVM (good spatial locality)
- Good temporal locality with respect to y

■ Disadvantages:

- Insertion into A is costly
- Poor temporal locality with respect to x

15

Impact of Matrix Sparsity on Performance

■ Addressing overhead (dense MVM vs. dense MVM in CSR):

- ~ 2x slower (example only)

■ Fundamental difference between MVM and sparse MVM (SMVM):

- Sparse MVM is input *dependent* (sparsity pattern of A)
- Changing the order of computation (blocking) requires changing the data structure (CSR)

16

Bebop/Sparsity: SMVM Optimizations

- **Idea:** Blocking for registers
- **Reason:** Reuse x to reduce memory traffic
- **Execution:** Block SMVM $y = y + Ax$ into micro MVMs
 - Block size $r \times c$ becomes a parameter
 - Consequence: Change A from CSR to $r \times c$ block-CSR (BCSR)
- **BCSR: Blackboard**

17

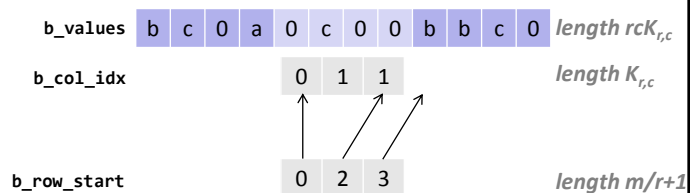
BCSR (Blocks of Size $r \times c$)

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

A as matrix ($r = c = 2$)

b	c		c
	a		
		b	b
		c	

A in BCSR ($r = c = 2$):



- **Storage:**
 - $rcK_{r,c}$ doubles + $(K_{r,c}+m/r+1)$ ints = $\Theta(rcK_{r,c})$
 - $rcK_{r,c} \geq K$

18

Sparse MVM Using 2 x 2 BCSR

```
void smvm_2x2(int bm, const int *b_row_start, const int *b_col_idx,
              const double *b_values, double *x, double *y)
{
    int i, j;
    double d0, d1, c0, c1;

    /* loop over bm block rows */
    for (i = 0; i < bm; i++) {
        d0 = y[2*i]; /* scalar replacement since reused */
        d1 = y[2*i+1];

        /* dense micro MVM */
        for (j = b_row_start[i]; j < b_row_start[i+1]; j++, b_values += 2*2) {
            c0 = x[2*b_col_idx[j]+0]; /* scalar replacement since reused */
            c1 = x[2*b_col_idx[j]+1];
            d0 += b_values[0] * c0;
            d1 += b_values[2] * c0;
            d0 += b_values[1] * c1;
            d1 += b_values[3] * c1;
        }
        y[2*i] = d0;
        y[2*i+1] = d1;
    }
}
```

19

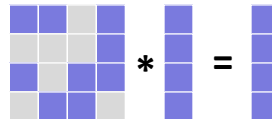
BCSR

■ Advantages:

- Temporal locality with respect to x and y
- Reduced storage for indexes

■ Disadvantages:

- Storage for values of A increased (zeros added)
- Computational overhead (also due to zeros)

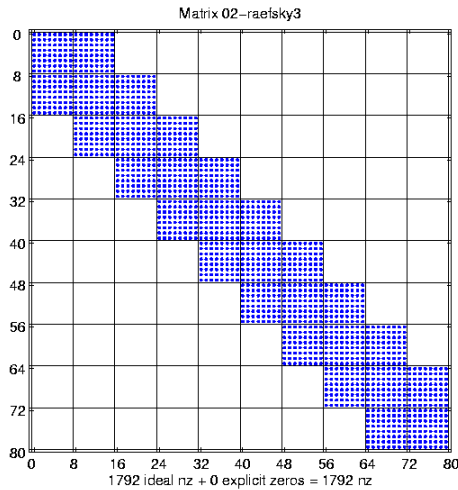


■ Main factors (since memory bound):

- **Plus:** increased temporal locality on x + reduced index storage
= reduced memory traffic
- **Minus:** more zeros = increased memory traffic

20

Which Block Size (r x c) is Optimal?

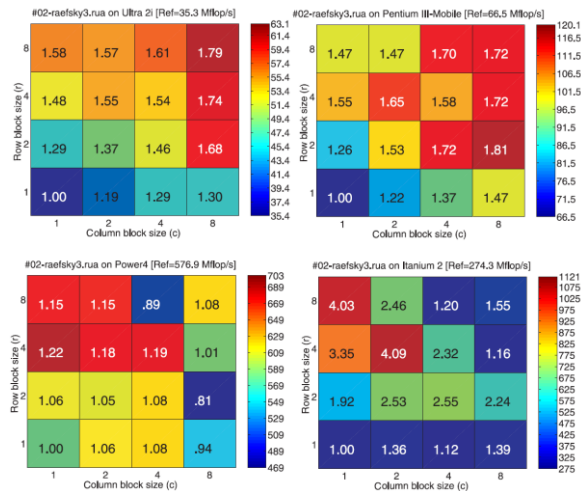


Example:

- 20,000 x 20,000 matrix (only part shown)
- Perfect 8 x 8 block structure
- No overhead when blocked r x c, with r, c divides 8

source: R. Vuduc, LLNL

Speed-up Through r x c Blocking



- machine dependent
- hard to predict

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

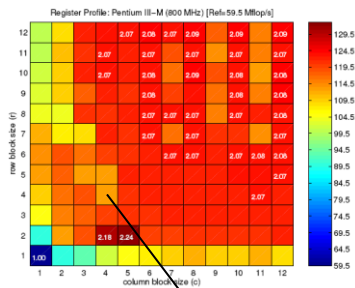
How to Find the Best Blocking for given A?

- Best block size is hard to predict (see previous slide)
- **Solution 1: Searching over all $r \times c$ within a range, e.g., $1 \leq r, c \leq 12$**
 - Conversion of A in CSR to BCSR roughly as expensive as 10 SMVMs
 - Total cost: 1440 SMVMs
 - Too expensive
- **Solution 2: Model**
 - Estimate the gain through blocking
 - Estimate the loss through blocking
 - Pick best ratio

23

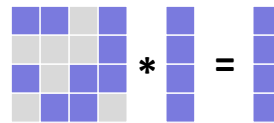
Model: Example

Gain by blocking (dense MVM)



1.4

Overhead (average) by blocking



$$16/9 = 1.77$$

$$1.4/1.77 = 0.79 \text{ (no gain)}$$

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

24

Model

- **Goal:** find best $r \times c$ for $y = y + Ax$

- **Gain** through $r \times c$ blocking (estimation):

$$G_{r,c} = \frac{\text{dense MVM performance in } r \times c \text{ BCSR}}{\text{dense MVM performance in CSR}}$$

dependent on machine, independent of sparse matrix

- **Overhead** through $r \times c$ blocking (estimation)

scan part of matrix A

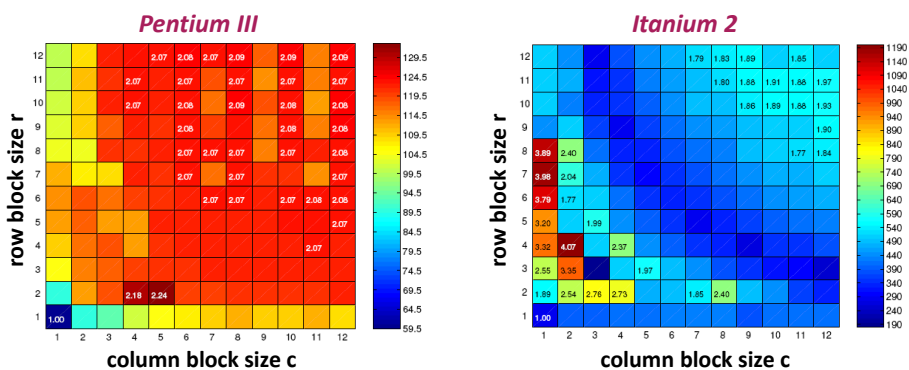
$$O_{r,c} = \frac{\text{number of matrix values in } r \times c \text{ BCSR}}{\text{number of matrix values in CSR}}$$

independent of machine, dependent on sparse matrix

- **Expected gain:** $G_{r,c}/O_{r,c}$

25

Gain from Blocking (Dense Matrix in BCSR)



- machine dependent
- hard to predict

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

Typical Result

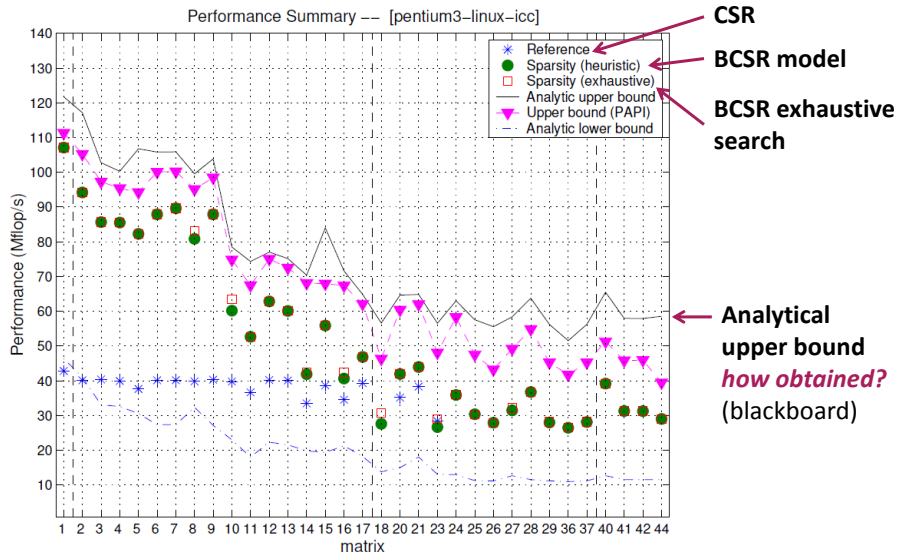


Figure: Eun-Jin Im, Katherine A. Yelick, Richard Vuduc. SPARSITY: An Optimization Framework for Sparse Matrix Kernels, *Int'l Journal of High Performance Comp. App.*, 18(1), pp. 135-158, 2004

Principles in Bebop/Sparsity Optimization

- **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):**
 - Unrolling + scalar replacement
- **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

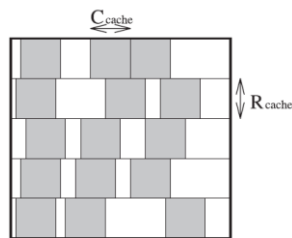
SMVM: Other Ideas

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

29

Cache Blocking

- Idea: divide sparse matrix into blocks of sparse matrices



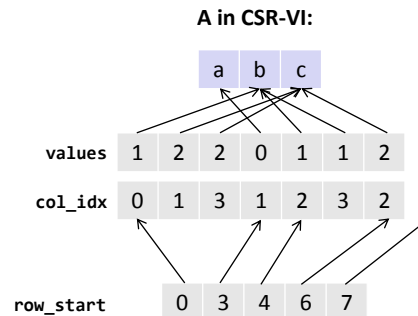
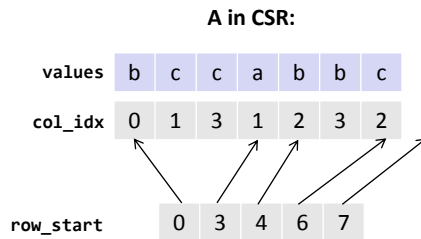
- Experiments:
 - Requires very large matrices (x and y do not fit into cache)
 - Speed-up up to 2.2x, only for few matrices, with 1 x 1 BCSR

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

Value Compression

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

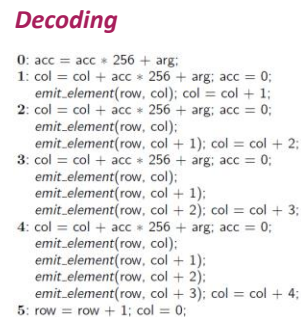
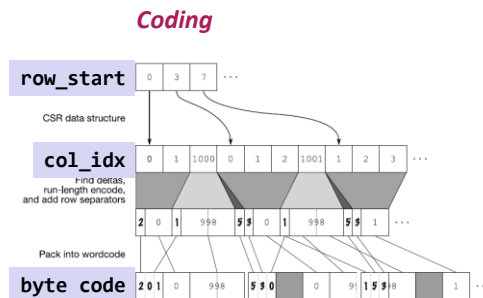
	b	c		c
		a		
			b	b
			c	



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



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	a	0	c	0	0	b	b	c	0
---	---	---	---	---	---	---	---	---	---	---	---

Values in 2 x 2 PBR

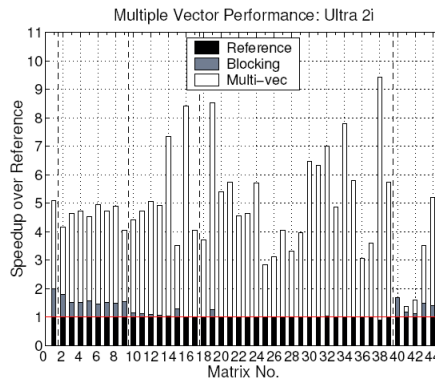
b	c	a	c	b	b	c
---	---	---	---	---	---	---

+ bit string: 1101 0100 1110

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

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