A structure-driven performance analysis of sparse matrix-vector multiplication

Prabhjot Sandhu, Clark Verbrugge, and Laurie Hendren

Sable Research Group
McGill University

23 April 2020

Outline

1. Introduction
2. Experimental Design
3. Research Questions : Effect of Matrix Structure
   - On the Choice of Storage Format
   - Within a Storage Format
   - Along with Hardware Characteristics
4. Summary and Future Work

Outline

Background : Sparse Matrix Storage Formats

- **A sparse matrix :** a matrix in which most of the elements are zero.
- **Basic sparse storage formats :**
  - Coordinate Format (COO)
  - Compressed Sparse Row Format (CSR)
  - Diagonal Format (DIA)
  - ELLPACK Format (ELL)

A

1	0	6	0
0	2	0	7
0	0	3	0
5	0	0	4

COO :

row	0	0	1	1	2	3	3
col	0	2	1	3	2	0	3
val	1	6	2	7	3	5	4

CSR :

row_ptr	0	2	4	5	7		
col	0	2	1	3	2	0	3
val	1	6	2	7	3	5	4

DIA :

data

-	-	-	5
1	2	3	4
6	7	-	-

offset

-3	0	2

ELL :

data

1	2	3	5
6	7	-	4

indices

0	1	2	0
2	3	-	3

Background : SpMV

Sparse Matrix-Vector Multiplication

- $y = Ax$, where A is a sparse matrix and the input vector x and output vector y are dense.
- Working set size : sizeof(A) + sizeof(x) + sizeof(y)

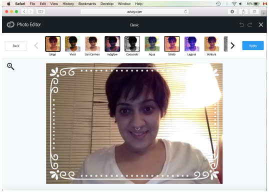
$$\underset{A}{\begin{bmatrix} 1 & 0 & 6 & 0 \\ 0 & 2 & 0 & 7 \\ 0 & 0 & 3 & 0 \\ 5 & 0 & 0 & 4 \end{bmatrix}} * \underset{x}{\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}} = \underset{y}{\begin{bmatrix} 7 \\ 9 \\ 3 \\ 9 \end{bmatrix}}$$

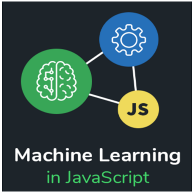
Why Sparse Matrices on the Web?

- Web-enabled devices everywhere!
- Various compute-intensive applications involving sparse matrices on the web.
  - Image editing
  - Computer-aided design
  - Text classification (data mining)
  - Deep learning
- Recent addition of WebAssembly to the world of JavaScript.

Why Sparse Matrices on the Web?

- Web-enabled devices everywhere!
- Various compute-intensive applications involving sparse matrices on the web.
  - Image editing
  - Computer-aided design
  - Text classification (data mining)
  - Deep learning
- Recent addition of WebAssembly to the world of JavaScript.





Why Sparse Matrices on the Web?

- Web-enabled devices everywhere!
- Various compute-intensive applications involving sparse matrices on the web.
  - Image editing
  - Computer-aided design
  - Text classification (data mining)
  - Deep learning
- Recent addition of WebAssembly to the world of JavaScript.

Why SpMV is so Important?

- A computational kernel used in many scientific and machine learning applications.
- occurs frequently in these applications.
- Hence, a good candidate for their performance optimization.

Why SpMV is so Important?

- A computational kernel used in many scientific and machine learning applications.
- occurs frequently in these applications.
- Hence, a good candidate for their performance optimization.

Why SpMV is so Important?

- A computational kernel used in many scientific and machine learning applications.
- occurs frequently in these applications.
- Hence, a good candidate for their performance optimization.

How to Optimize SpMV Performance

1. Select an optimal format to store the input sparse matrix.
2. Apply data and low-level code optimizations to a single format.

Depends on the structure of the matrix and the machine characteristics.

How to Optimize SpMV Performance

1. Select an optimal format to store the input sparse matrix.
2. Apply data and low-level code optimizations to a single format.

Depends on the structure of the matrix and the machine characteristics.

How to Optimize SpMV Performance

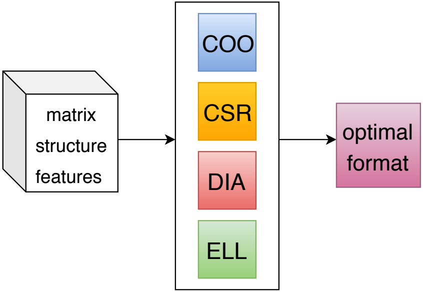
1. Select an optimal format to store the input sparse matrix.
2. Apply data and low-level code optimizations to a single format.

Depends on the structure of the matrix and the machine characteristics.

Our Goal

To understand the effect of :

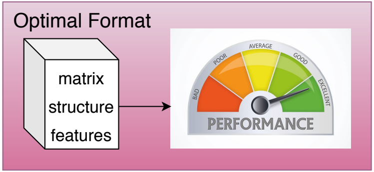
1. matrix structure on the choice of storage format.
2. matrix structure on the SpMV performance within a storage format.
3. interaction between matrix structure and hardware characteristics on the SpMV performance.



Our Goal

To understand the effect of :

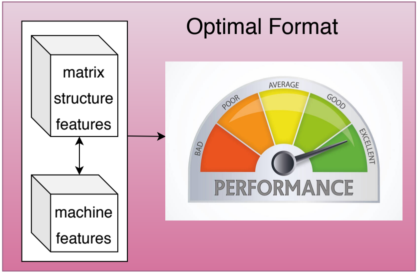
1. matrix structure on the choice of storage format.
2. matrix structure on the SpMV performance within a storage format.
3. interaction between matrix structure and hardware characteristics on the SpMV performance.



Our Goal

To understand the effect of :

1. matrix structure on the choice of storage format.
2. matrix structure on the SpMV performance within a storage format.
3. interaction between matrix structure and hardware characteristics on the SpMV performance.



Outline

Reference Implementations and Measurement Setup

Developed a reference set of sequential C and hand-tuned WebAssembly implementations of SpMV for different formats on same algorithmic lines.

```
void spmv_coo(int *row, int *col, float *val, int nnz, int N, float *x, float *y)
{ int i;
  for(i = 0; i < nnz ; i++)
    y[row[i]] += val[i] * x[col[i]];
}
```

Listing 1: Single-precision SpMV COO implementation in C

- **Benchmarks** : Around 2000 real-life sparse matrices from The SuiteSparse Matrix Collection.
- **Sparse Storage Formats** : COO, CSR, DIA, ELL
- Measured SpMV Performance for C and WebAssembly in FLOPS (Floating point operations per second).

Target Languages and Runtime

- **Machine Architecture**

Intel Core i7-3930K with 6 3.20GHz cores, 12MB last-level cache and 16GB memory,running Ubuntu Linux 16.04.2

- **C**

Compiled with gcc version 7.2.0 at optimization level -O3

- **WebAssembly**

Used Chrome 74 browser (Official build 74.0.3729.108 with V8 JavaScript engine 7.4.288.25) as the execution environment with –experimental-wasm-simd flag to enable the use of SIMD instructions.

How we chose the optimal format?

x%-affinity

We say that an input matrix A has an x%-affinity for storage format F, if the performance for F is at least x% better than all other formats and the performance difference is greater than the measurement error.

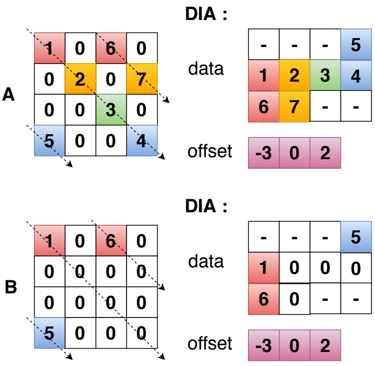
Example

For example, if input array A in format CSR, is more than 10% faster than input A in all other formats, and 10% is more than the measurement error, then we say that A has a 10%-affinity for CSR.

Outline

Matrix Structure Feature : *dia_ratio*

- $dia\_ratio = \dfrac{ndiag\_elems}{nnz}$ where, **nnz** : number of non-zeros, **ndiag_elems** : number of elements in the diagonals
- Indicates if the given matrix is a good fit for DIA format or not.
- dia_ratio(A) = 7/7 = 1
- dia_ratio(B) = 7/3 = 2.33



DIA Format

Matrices with dia_ratio <= 3 show affinity towards the DIA format, except for a few matrices.

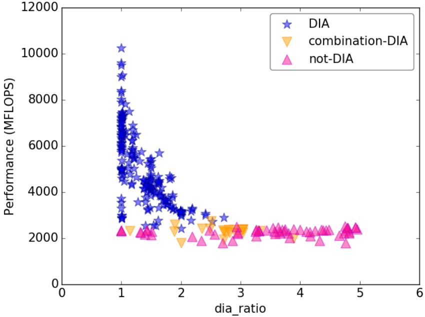


Figure: C

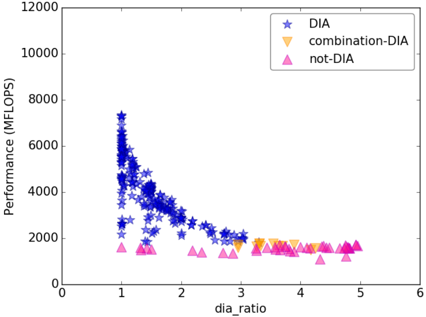


Figure: Wasm

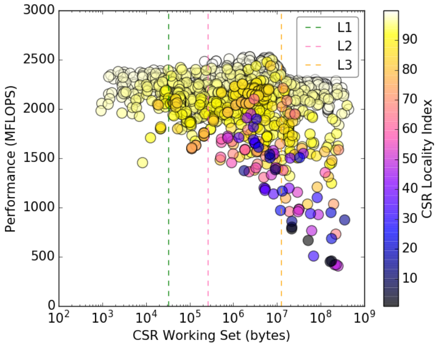
Relationship between Storage Format and Structure Features

Format	Feature(s)	Priority
DIA	*dia_ratio* $\leq$ 3 and large N	1
ELL	*ell_ratio* $\simeq$ 1 and small *max_nnz_per_row*	2
COO	nnz $<$ N or small *avg_nnz_per_row* and uneven number of non-zeros per row	3

Outline

SpMV Performance within CSR Matrices

- CSR Working Set : (N+1) + 2*nnz + 2*N
- Irregular access for vector x affects performance.
- Introduced some new matrix structure features : *ELL Locality Index, CSR Locality Index*
- Based on data locality model
- Using reuse-distance concept



CSR Locality Index

indicator of irregular memory access for vector x for a CSR matrix.

CSR Locality Index : Step 1

- Calculate Row Reuse Distance for each non-zero.
- Row Reuse Distance (rrd) : Distance from the last non-zero whose column index corresponds to the same cache line of the input vector x.
- Unit of distance : rows

*Assume the cache line size to be 2 and cache size to be fixed for this example.

A

1	0	6	0
0	2	0	7
0	0	3	0
5	0	0	4

CSR :

row_ptr	0	2	4	5	7		
col	0	2	1	3	2	0	3
val	1	6	2	7	3	5	4

CSR Locality Index : Step 1

- Calculate Row Reuse Distance for each non-zero.
- Row Reuse Distance (rrd) : Distance from the last non-zero whose column index corresponds to the same cache line of the input vector x.
- Unit of distance : rows

*Assume the cache line size to be 2 and cache size to be fixed for this example.

A

1	0	6	0
0	2	0	7
0	0	3	0
5	0	0	4

x

1	cache line 1
1	
1	cache line 2
1	

CSR :

row_ptr	0	2	4	5	7		
col	0	2	1	3	2	0	3
val	1	6	2	7	3	5	4
x-vector access pattern	x[0]	x[2]	x[1]	x[3]	x[2]	x[0]	x[3]
rrd	-	-	1	1	1	2	1

CSR Locality Index : Step 2

- Calculate CSR Reuse Distance using frequency distribution over Row Reuse Distance (rrd).
- CSR Reuse Distance[p] : the number of non-zeros of sparse matrix A stored in the CSR format which access the input vector x with p Row Reuse Distance.

CSR :

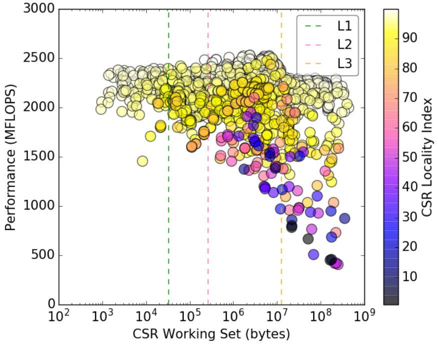
row_ptr	0	2	4	5	7		
col	0	2	1	3	2	0	3
val	1	6	2	7	3	5	4
x-vector access pattern	x[0]	x[2]	x[1]	x[3]	x[2]	x[0]	x[3]
rrd	-	-	1	1	1	2	1

Index	0	1	2	3
Frequency	0	4	1	0

CSR Locality Index : Step 3

- Calculate CSR Locality Index using cumulative percentage over CSR Reuse Distance.
- $\mathit{CSR\ Locality\ Index} = \dfrac{\sum_{p=0}^{15} \mathit{CSR\ Reuse\ Distance}[p]}{nnz} \times 100$
- This feature accounts for :
  - spatial locality for the non-zeros in a row.
  - temporal locality for the non-zeros in the neighbouring rows.

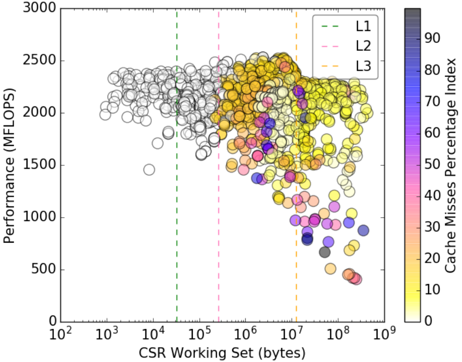
* We chose the limit to be 15 based on our experiments



Outline

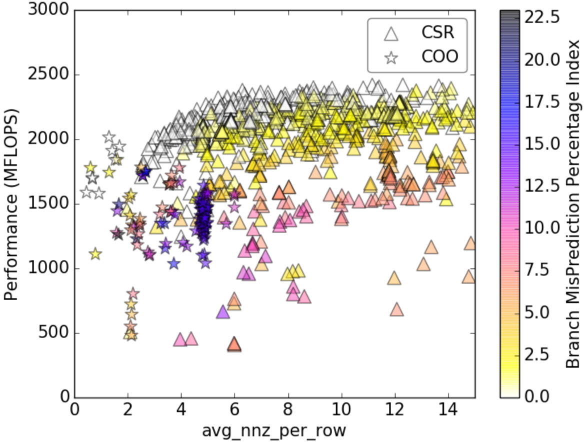
Cache Memory : CSR Performance

- Features based on data locality model have their roots in the hardware features like data cache misses.
- Measured true performance counters using PAPI tool.
- $Index = \frac{PAPI\_L1\_DCM \vee PAPI\_L2\_DCM \vee PAPI\_L3\_TCM}{nnz} \times 100$



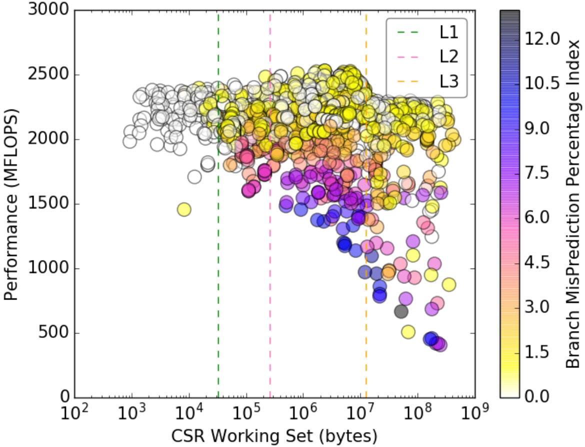
Branch Prediction Unit : CSR vs COO

- $Index = \frac{PAPI\_BR\_MSP}{PAPI\_BR\_PRC + PAPI\_BR\_MSP} \times 100$



Branch Prediction Unit : CSR Performance

- $Index = \frac{PAPI\_BR\_MSP}{PAPI\_BR\_PRC + PAPI\_BR\_MSP} \times 100$



Outline

Summary

- The optimal choice of storage format is governed both by the structure of the matrix and the code optimization opportunities available.
- Due to different code generation strategy, the SpMV performance suffers in the case of WebAssembly for Chrome (v8) browser.
- Our data locality based structure features estimate if the SpMV performance is affected by the irregular memory accesses for vector x.
- We validate our evaluations and parameter choices using hardware performance counters.

Future Work

- Further explore to quantify the impact of additional hardware features on SpMV performance via matrix structure features.
- Explore new optimization opportunities for hand-tuned WebAssembly implementations through the upcoming WebAssembly instructions.
- Develop parallel versions of SpMV based on multithreading features like web workers.
- Develop automatic techniques to choose the best format for web-based SpMV.

Contact details

name : Prabhjot Sandhu
e-mail : prabhjot.sandhu@mail.mcgill.ca
webpage : `https://www.cs.mcgill.ca/~psandh3`