HICREST

UNITRENTO

Introduction to Parallel Computing

a.y. 2025 – 2026

Flavio Vella

Deliverable Presentation and Timeline

Sparse data-format and sequential SpmV

Performance models vs Programming models

- The Matrix-Vector product is a special case of general matrix multiplication (GEMM), where the second operand is a vector.

- This means the output of $\mathrm{M} \cdot \vec{v}$ is a vector $\vec{c}$ such that:

$$\vec{c}_i = < M_{i,\cdot}, \vec{v} > = \sum_{j=1}^{m} M_{i,j} \cdot \vec{v}_j$$

0	**1**	**2**	**3**
10	**11**	**12**	**13**
20	**21**	**22**	**23**
30	**31**	**32**	**33**

✖

1
1
1
1

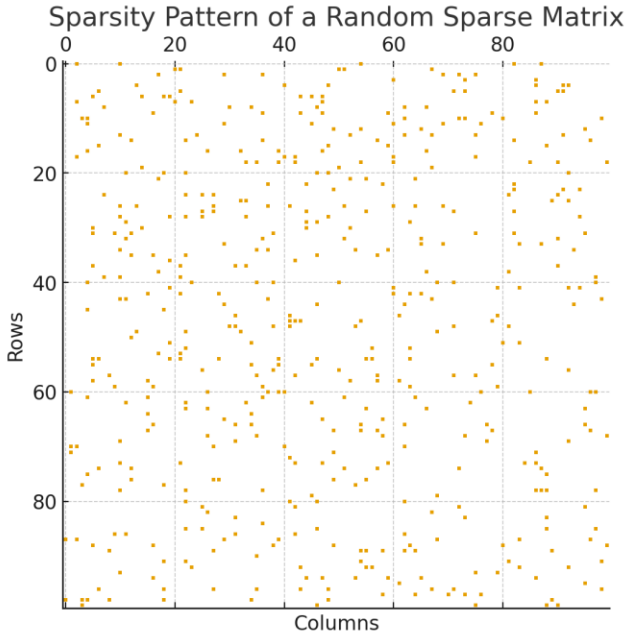
=

6
46
86
126

What is a sparse Matrix ?

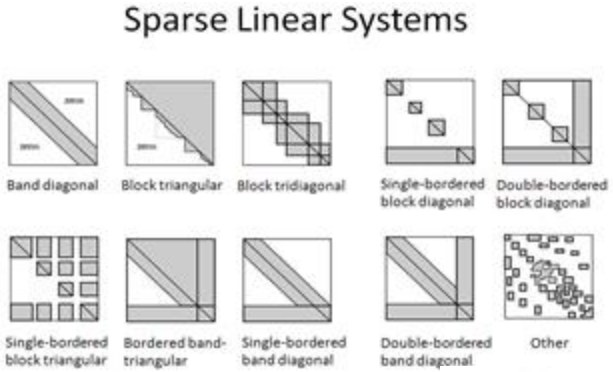
- The **Sparse** Matrix-Vector is a matrix-vector multiplication where the matrix operand is sparse (typically more than 50% of the entries are zeros)

A sparse matrix is a matrix where the number of nonzero elements is much smaller than the total number of elements (nnz << rows x cols).

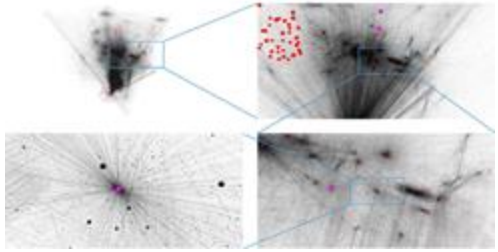


Applications and sparsity patterns

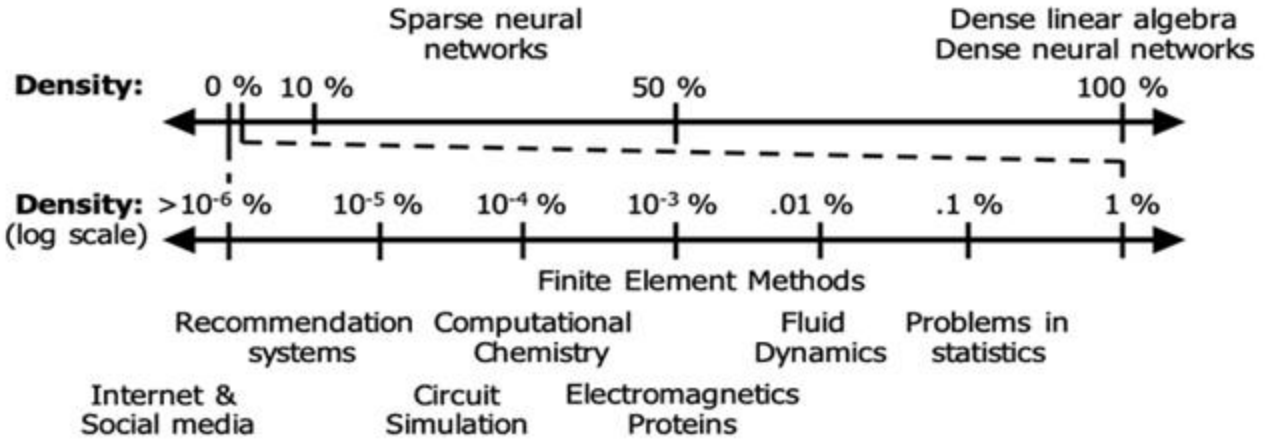
- A **sparsity pattern** of a matrix formally describes the locations of its nonzero elements, independent of their actual numerical values.
- The world is sparse, from science to AI, matrices exhibit different sparsity patterns and degrees of sparsity.



Credit Prof. Peter Bermel



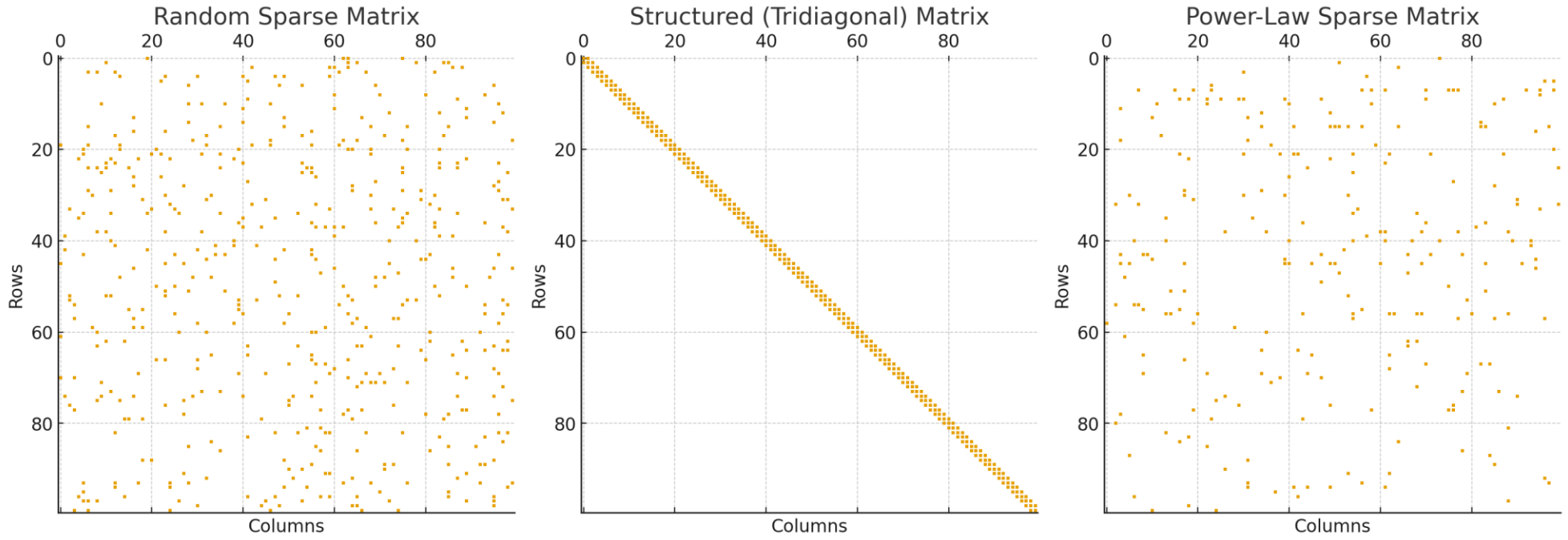
Albert-László Barabási: Network Science



Kartik Hegdem Edgar Solomonik, et al. ExTensor: An Accelerator for Sparse Tensor Algebra. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52). Association for Computing Machinery, New York, NY, USA, 319–333. https://doi.org/10.1145/3352460.3358275

Sparse Matrix issues

- Storing in the dense format is not space-efficient
- Different sparsity patterns imply different access – often unpredictable access pattern to the memory



Storage

- Sparse matrices can be stored efficiently in different ways (storage formats).
- The simplest one is the **coordinate matrix format (COO)**.

	0.5			1.2	
		-1.0			
2.3					0.2
			4.5		
	-3.0			6.1	

Given a sparse matrix A, we represent it with three vectors:

1. The first two arrays will store the coordinate of each nnz value:

`Arow`:	0	0	1	2	2	4	5	5
`Acol`:	1	4	2	0	5	3	1	4

2. The third array will store all the values:

`Aval`:	0.5	1.2	-1.0	2.3	0.2	4.5	-3.0	6.1

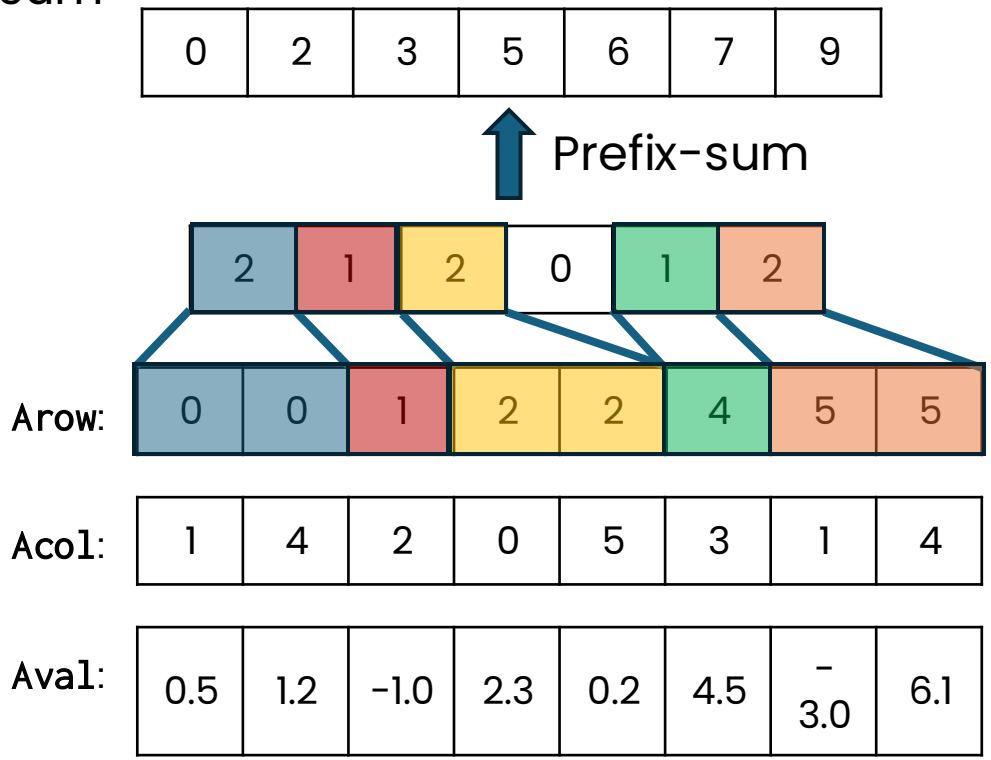
Compressed sparse-row

COO storage store multiple times the row index.

We can do better by using one index for each row. performs two operations:

1. **Sort elements** according to their coordinate (element $(\mathrm{i},\mathrm{j}) < (\mathrm{i}',\mathrm{j}') \leftrightarrow (i < i')\ or\ (i = i' and\ j < j')$)
2. **Compress the row pointer** with a prefix-sum

	0.5			1.2	
		-1.0			
2.3					0.2
			4.5		
	-3.0			6.1	

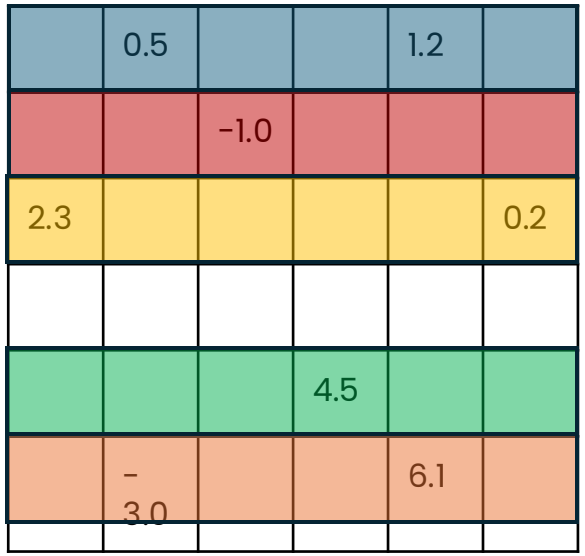


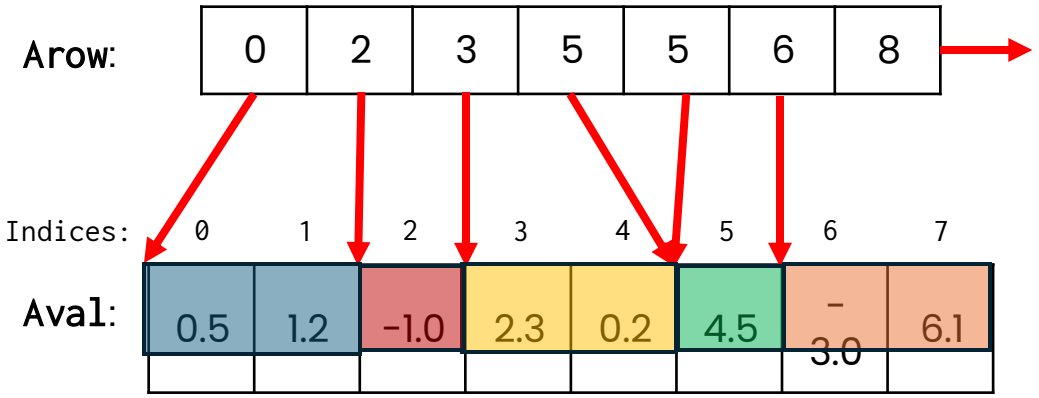
Compressed sparse-row

The CSR row-array stores the start index of the elements related to a specific row

```
for(int i = 0; i<nrows; i++) {
     for(int j = row_ptr[i]; i<row_ptr[i+1]; j++) {
              fprintf(stdout, “element (%d,%d) has value %f\n”, i, col[j], val[j]):
      }
}
```

A





Deliverable Instructions

UNITRENTO

- **Deadline update! 17-11-2025**
- **Develop** a code:
  1. Read a matrix in the Matrix Format and convert to CSR
  2. Multiply the Matrix with a randomly generated array (sequential code)
  3. Design a parallel code on a shared-memory system (e.g., using OpenMP)
- **Benchmarking**:
  - Report the results of the multiplication in Milliseconds (at least 10 runs, report the 90% percentile)
  - Select at least five matrices with different degrees of sparsity
  - Sequential code vs Parallel Code by increasing the number of threads
  - Evaluate different scheduling strategies
  - Identify the bottleneck
- **Bonus:** try to optimise the code by exploring the scientific papers.

Disclaimer

- Further information will be given in the next lectures
- **How to Profile Sequential Code** and the **best practice of benchmarking** will also be discussed!

Exercise 1

UNITRENTO

1. Write a sequential SpMV algorithm that multiplies a randomly generated COO with a dense vector where all the entrances are set to 1.
2. Do the same by using a CSR data format
3. Compare the runtime performance of the COO vs the CSR implementation.
4. Profile the cache's behaviour with valgrind (next lab)

Supplementary exercise:

- Download the matrix marker file from Suitsparse (https://sparse.tamu.edu/) and read the sparse matrix from the file.

Parse matrix format (mtx extension) example:

https://math.nist.gov/MatrixMarket/mmio/c/example_read.c

1. wget https://suitesparse-collection-website.herokuapp.com/MM/HB/1138_bus.tar.gz
2. gzip -d 1138_bus.tar.gz
3. tar -xf 1138_bus.tar
4. ls 1138_bus/1138_bus.mtx

References

- Samuel Williams, Leonid Oliker, Richard Vuduc, John Shalf, Katherine Yelick, and James Demmel. 2007. Optimization of sparse matrix-vector multiplication on emerging multicore platforms. In Proceedings of the 2007 ACM/IEEE conference on Supercomputing (SC '07). Association for Computing Machinery, New York, NY, USA, Article 38, 1–12. https://doi.org/10.1145/1362622.1362674
- Aydin Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. 2009. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures (SPAA '09). Association for Computing Machinery, New York, NY, USA, 233–244. https://doi.org/10.1145/1583991.1584053
- Other references look at Google Scholar: “Sparse Matrix Vector multiplication on OpenMP” or “Multicore Architectures”

Storage format:

- NVIDIA https://docs.nvidia.com/nvpl/latest/sparse/storage_format/sparse_matrix.html
- Intel https://www.intel.com/content/www/us/en/docs/onemkl/developer-reference-dpcpp/2024-0/sparse-storage-formats.html