

CS601: Software Development for Scientific Computing

Autumn 2022

Week4: Build tool (Make demo), Motifs –
Matrix Computations with Dense Matrices

make – Recap and Demo

- Minimal build
 - What if only `scprod.cpp` changes?
- Special targets (`.phony`)
 - E.g. explicit request to `clean` executes the associated recipe. What if there is a file named `clean`?
- Organizing into folders
 - Use of variables (built-in (`CXX`, `CFLAGS`) and automatic (`$@`, `^`, `<`))

refer to week3_codesamples

Recall Motifs from Week1

Scientific Software - Motifs

noun

1. a decorative image or design, especially a repeated one forming a pattern.
"the colourful hand-painted motifs which adorn narrowboats"

Similar:

design

pattern

decoration

figure

shape

logo

monogram



2. a dominant or recurring idea in an artistic work.
"superstition is a recurring motif in the book"

- | | |
|---------------------------|--------------------------------|
| 1. Finite State Machines | 8. Dynamic Programming |
| 2. Combinatorial | 9. <u>N-Body (/ particle)</u> |
| 3. Graph Traversal | 10. MapReduce |
| 4. <u>Structured Grid</u> | 11. Backtrack / B&B |
| 5. <u>Dense Matrix</u> | 12. Graphical Models |
| 6. <u>Sparse Matrix</u> | 13. <u>Unstructured Grid</u> |
| 7. <u>FFT</u> | |

Matrix Algebra and Efficient Computation

- Pic source: the Parallel Computing Laboratory at U.C. Berkeley: A Research Agenda Based on the Berkeley View (2008)

<i>Motif</i>	Embed	Desktop	Games	DB	ML	HPC	Medicine	Music	Speech	CBIR	Browser	Motif	Desktop	Games	DB	ML	HPC	Medicine	Music	Speech	CBIR	Browser		
1 Finite State Mach.												9 N-Body												
2 Combinational												10 MapReduce												
3 Graph Traversal												11 Backtrack/B&B												
4 Structured Grid												12 Graphical Models												
5 Dense Matrix												13 Unstructured Grid												
6 Sparse Matrix												<i>Temperature Chart of Need</i>				DB = database								
7 Spectral (FFT)												Hot	Warm	Med	Cool	ML = machine learning								
8 Dynamic Prog																	HPC = High Perf. Comp.							

Figure 4. Temperature Chart of the 13 Motifs. It shows their importance to each of the original six application areas and then how important each one is to the five compelling applications of Section 3.1. More details on the motifs can be found in (Asanovic, Bodik et al. 2006).

Matrix Multiplication

- Why study?
 - An important “kernel” in many linear algebra algorithms
 - Most studied kernel in high performance computing
 - Simple. Optimization ideas can be applied to other kernels
- Matrix representation
 - Matrix is a 2D array of elements. Computer memory is inherently linear
 - C++ and Fortran allow for definition of 2D arrays. 2D arrays stored row-wise in C++. Stored column-wise in Fortran. E.g.
`// stores 10 arrays of 20 doubles each in C++`
`double** mat = new double[10][20];`

Storage Layout - Example

- Matrix (**2D**): $A = \begin{bmatrix} A(0,0) & A(0,1) & A(0,2) \\ A(1,0) & A(1,1) & A(1,2) \\ A(2,0) & A(2,1) & A(2,2) \end{bmatrix}$

$A(i, j) = A(\text{row}, \text{column})$ refers to the matrix element in the i^{th} row and the j^{th} column

- Row-wise (/Row-major) storage in memory:

$A(0,0)$	$A(0,1)$	$A(0,2)$	$A(1,0)$	$A(1,1)$	$A(1,2)$	$A(2,0)$	$A(2,1)$	$A(2,2)$
----------	----------	----------	----------	----------	----------	----------	----------	----------

- Column-wise (/Column-major) storage in memory:

$A(0,0)$	$A(1,0)$	$A(2,0)$	$A(0,1)$	$A(1,1)$	$A(2,1)$	$A(0,2)$	$A(1,2)$	$A(2,2)$
----------	----------	----------	----------	----------	----------	----------	----------	----------

- Generalizing data storage order for ND:** last index changes fastest in row-major. Last index changes slowest in col-major.

Storage Layout - Exercise

- For a 3D array (tensor) assume $A(i, j, k) = A(\text{row}, \text{column}, \text{depth})$



- What is the offset of $A(1, 2, 1)$? as per row-major storage?
- What is the offset of $A(1, 2, 1)$? as per col-major storage?