CS601: Software Development for Scientific Computing

Autumn 2022

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

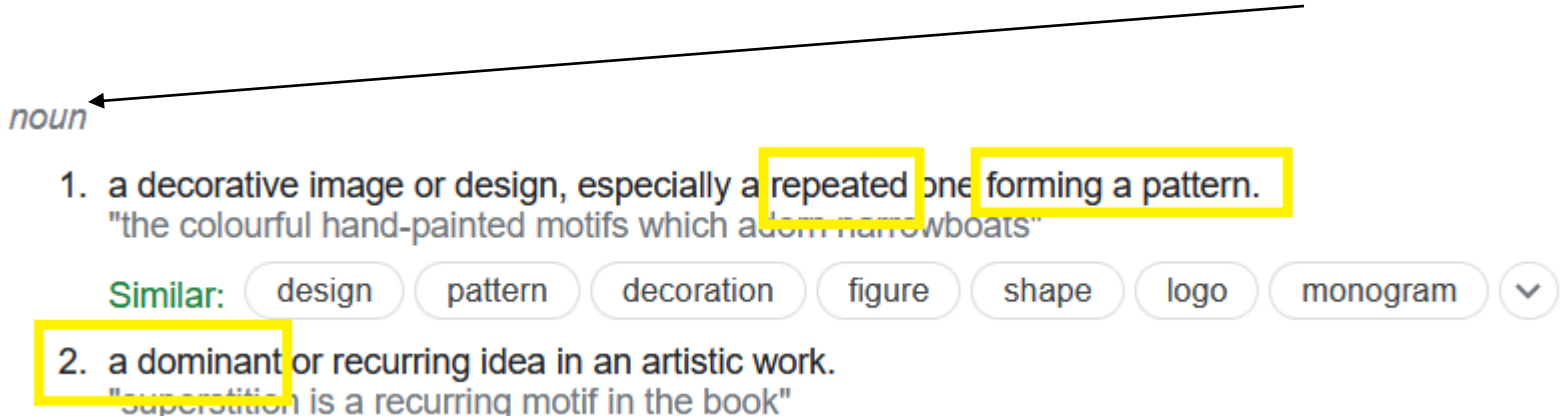
`make` – Recap and Demo

- 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



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

Matrix Algebra and Efficient Computation

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

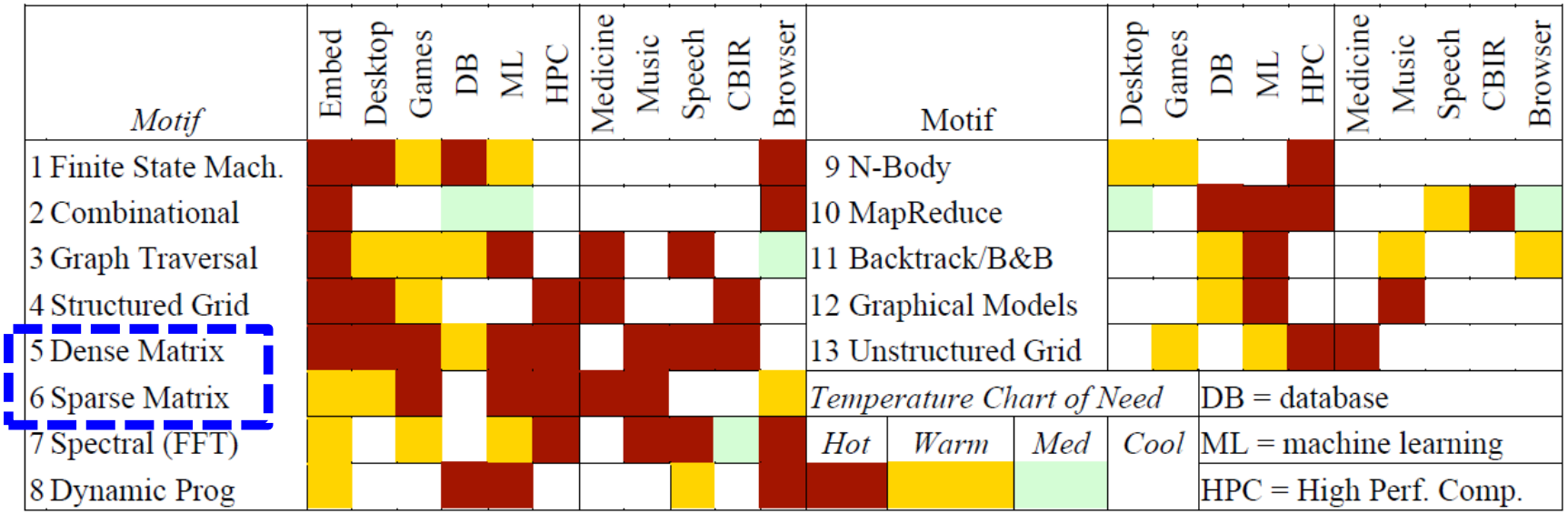


Figure 4. Temperature Chart of the 13 Motifs. It shows their importance to each of the original six application areas and then how important each one is to the five compelling applications of Section 3.1. More details on the motifs can be found in (Asanovic, Bodik et al. 2006).

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

$A(i,j) = A(row, column)$ refers to the matrix element in the ith row and the jth 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 $A(i,j,k) = A(row, column, depth)$

- 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?