Row-major order and column-major order

widely-used strategies to store multidimensional arrays in linear storage

0	1	2
3	4	5

A_{00}	A_{01}	A_{02}
A_{10}	A_{11}	A_{12}

Note: zero-indexing

E. W. Dijkstra, "Why numbering should start at zero" (1982)

Row-major order

0	1	2	3	4	5

Column-major order

0	3	1	4	2	5

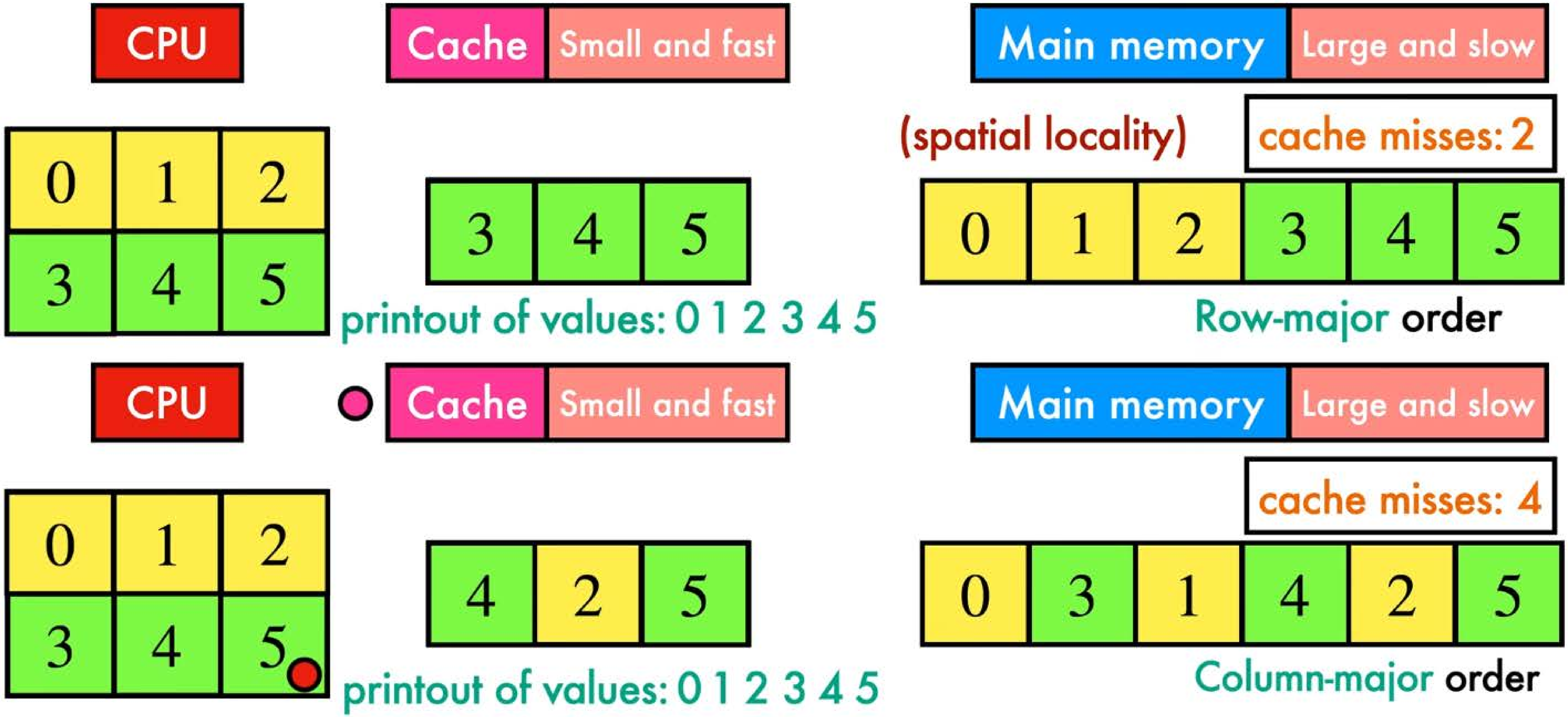
Lexicographic order

A_{00}	A_{01}	A_{02}	A_{10}	A_{11}	A_{12}

Colexicographic order

A_{00}	A_{10}	A_{01}	A_{11}	A_{02}	A_{12}

Why is storage order important? Caching



Memory address formulas

0	1	2
3	4	5

i

j

$\text{rows} = 2$

$\text{cols} = 3$

Row-major order:

0	1	2	3	4	5

$\text{index} = 3$

$$\text{index} = i \cdot \text{cols} + j$$

$i = 1$ | $j = 0$ | $\text{index} = 1 \cdot 3 + 0$

Column-major order:

0	3	1	4	2	5

$\text{index} = 1$

$$\text{index} = j \cdot \text{rows} + i$$

$i = 1$ | $j = 0$

Higher dimensions

Suppose a D-dimensional tensor with shape $N_0 \times N_1 \times \ldots \times N_{D-1}$

dim d is indexed by i_d | $i_d \in \{0, \ldots, N_d - 1\}$ | $d \in \{0, \ldots, D - 1\}$

Row-major order: $\text{index} = \boxed{i_{d-1}} + N_{d-1} \cdot (i_{d-2} + N_{d-2} \cdot (\ldots + N_1 i_0))$

Last dimension is contiguous ("moves fastest")

Column-major order: $\text{index} = \boxed{i_0} + N_0 \cdot (i_1 + N_1 \cdot (\ldots + N_{d-2} i_{d-1}))$

Zeroth dimension is contiguous

Conventions in different languages

<u>Row-major order:</u>

C
C++
Pascal
<u>ML Libraries</u>
PyTorch
TensorFlow

<u>Column-major order:</u>

FORTRAN
R
MATLAB
Julia