Prof. Dr. D. Kranzlmüller, Dr. K. Fürlinger

Parallel Computing

WS 2017/18

Session 3: Optimizing DGEMM

Tobias Fuchs, M.Sc.
tobias.fuchs@nm.ifi.lmu.de

Today's session:

Preparation for the next assignment (online later today):

„Optimize a given naïve implementation of Matrix-Matrix multiplication"

- This is a "project assignment" due in ***3 weeks.***

Today's session:

Preparation for the next assignment (online later today):

„Optimize a given naïve implementation of Matrix-Matrix multiplication"

- This is a "project assignment" due in ***3 weeks.***
- ***It is unfathomably relevant to the final exam.***
- You may work in groups but we really don't recommend freeloading. Really learn these concepts. ***Breathe*** them.
- There will still be a new 1-week assignment next week.

Schedule:

Session 3	today	Preparation for the 3-week assignment (DGEMM)
Session 4		Discussing your progress in the DGEMM-assignment Preparation for new assignment
Session 5		Solutions to session 4 assignment

Today's session:

Preparation for the next assignment (online later today):

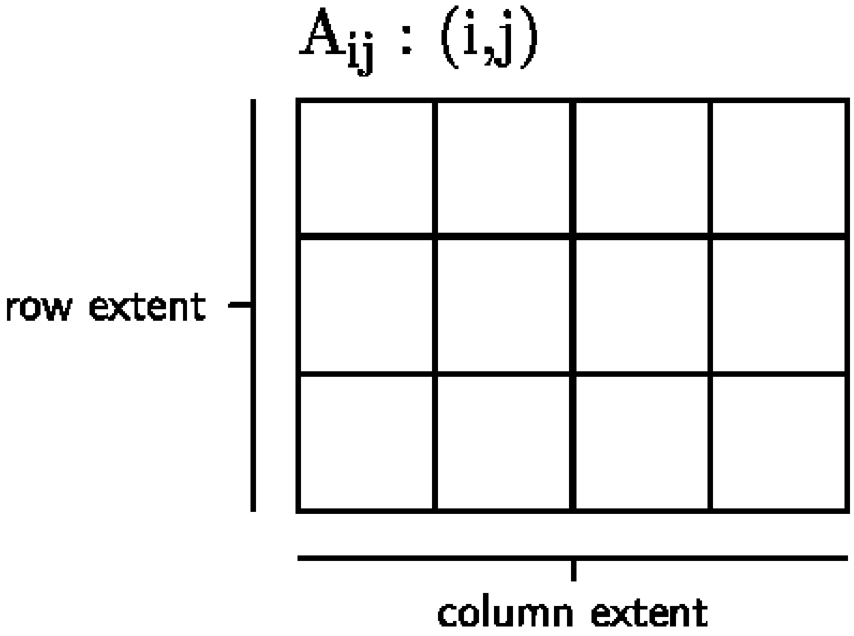
- Fundamentals of DGEMM
  (Dense General Matrix-Matrix Multiplication)
- Cache and locality
- Other optimization techniques like loop unrolling
- Restricting to sequential variant for now

Recap: Matrices

Matrices

A matrix is a rectangular array of values arranged in rows and columns (d'uh).

An $n \times m$ matrix has … ?

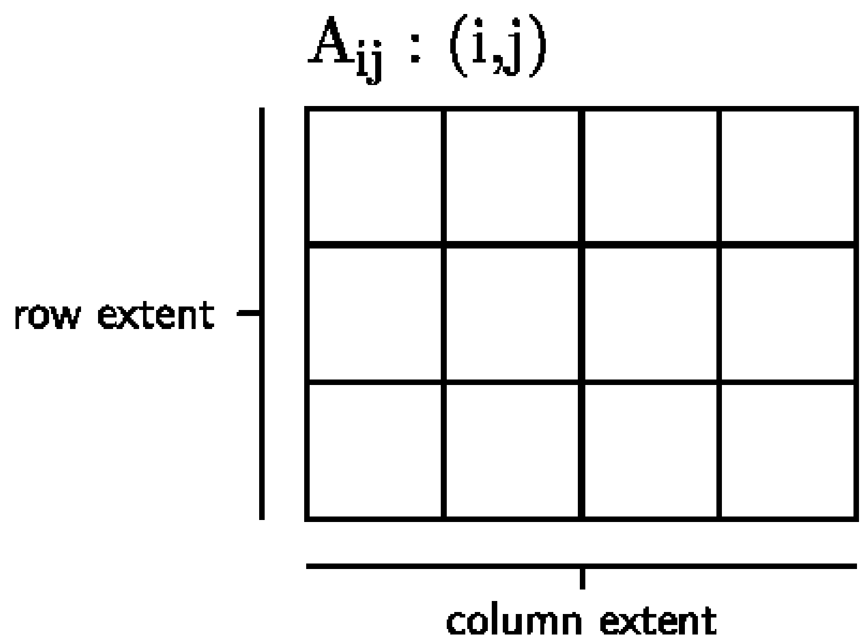


Matrices

A matrix is a rectangular array of values arranged in rows and columns (d'uh).

An $n \times m$ matrix has
n rows
m columns

Here:
3×4 matrix



Indices in rectangular arrays

Matrix notation

A_{ij} : (i,j)

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4

Cartesian notation

A_{xy} : (x,y)

1,1	2,1	3,1	4,1
1,2	2,2	3,2	4,2
1,3	2,3	3,3	4,3

Memory Storage Order

not to be confused with *memory ordering*,
i.e. the CPUs ability to reorder memory operations

Memory Storage Order

A matrix has two logical dimensions.

Memory, however is linear (one physical dimension).

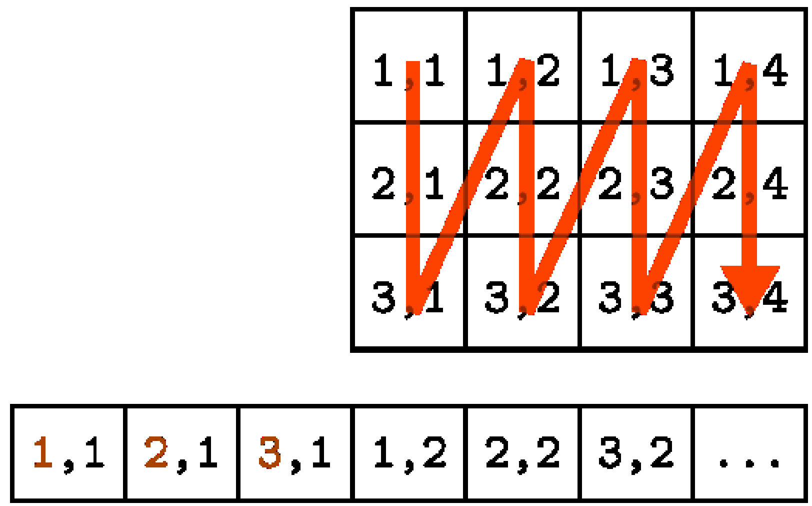
Memory order describes how multi-dimensional values are stored in linear memory.

Memory Storage Order

Column-major

Index in **left-most** dimension moves faster.

This is the Fortan style.

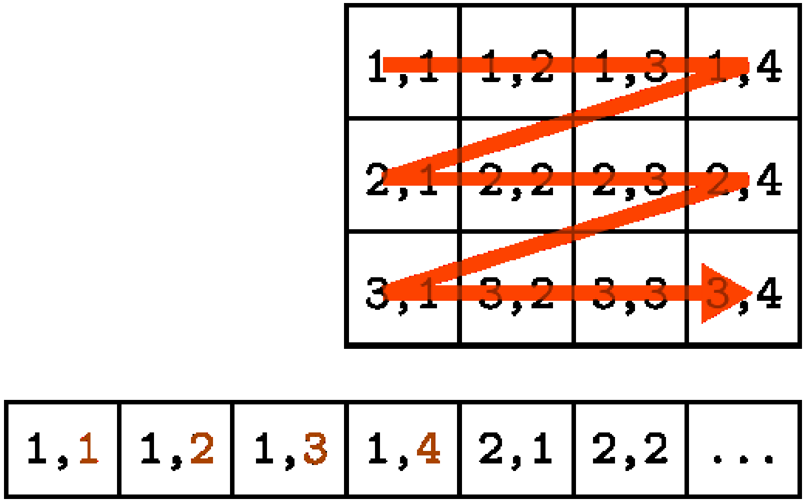


Memory Storage Order

Row-major

Index in **right-most** dimension moves faster.

This is the C style.




```
int M[N][M]
typeof(M[0]) -> int[M]
```

i.e. `M[i]` points to an array of row values (see C intro)

Matrix Product
Basics and Optimization

Matrix Product Basics

- Matrices are *arrays of numbers.*
- Different from elemental numbers (integer, complex, …), there is no unique way to define "*the*" multiplication of matrices.
- "Matrix multiplication" may refer to:

Hadamard product	entry-wise, like addition
Kronecker product	outer product, block-matrix
Matrix product	the one you know from school

Matrix Product Basics

$$C = AB$$

A: $n \times m$ matrix
B: $m \times p$ matrix
C: ?

Matrix Product Basics

$$C = AB$$

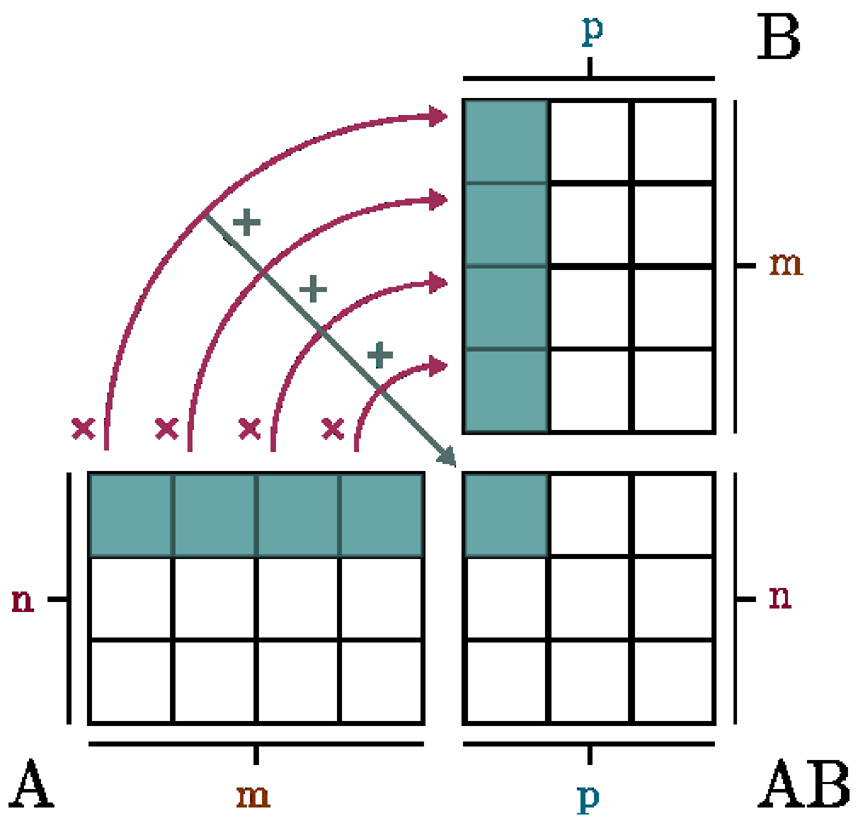
A: $n \times m$ matrix
B: $m \times p$ matrix
C: $n \times p$ matrix

Matrix Product Basics

$$(AB)_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

Notation in summation convention:

$$(AB)_{ij} = A_{ik} B_{kj}$$



Matrix Product – Naïve Implementation

```
for (index i = 0; i < n; i++) {
    for (index j = 0; j < p; j++) {
        double sum = 0.0;
        for (index k = 0; k < m; k++) {
            sum += A[i][k] * B[k][j];
        }
        C[i][j] = sum;
    }
}
```

Discuss with respect to memory layout of the matrices.

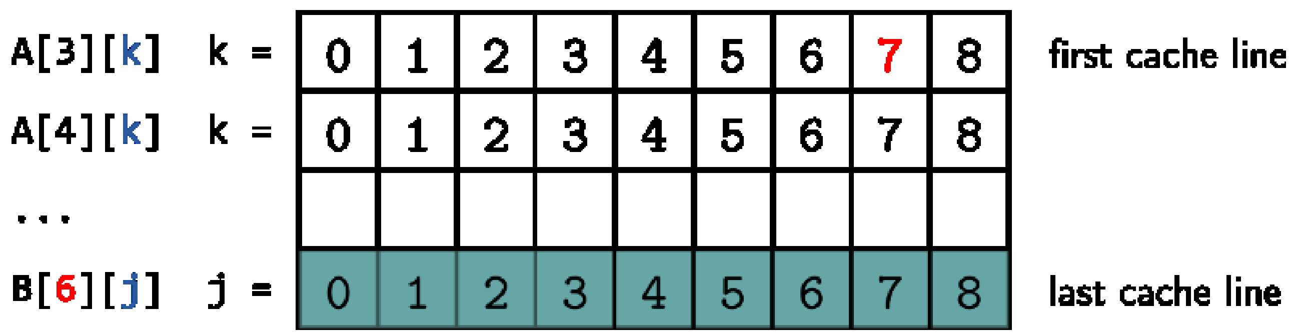
Matrix Product – Naïve Implementation

```
for (index i = 0; i < n; i++) {
    for (index j = 0; j < p; j++) {
        double sum = 0.0;
        for (index k = 0; k < m; k++) {
            sum += A[i][k] * B[k][j];
        }
        C[i][j] = sum;
    }
}
```

How about loop unrolling?

Considering Cache Locality

Arrays are contiguous memory blocks, so large chunks of them will be loaded into the cache upon first access.



Considering Cache Locality

- Cache locality matters a ***lot***.
- Loading data from main memory into cache takes ***hundreds*** of CPU cycles
- Cache misses dominate running time more than the actual calculations
- In an upcoming session, I will present tools to actually measure performance, e.g. using cache miss counters.

Lessons Learned for Optimization

- Make yourself familiar with blocking / tiling optimization techniques (you will find lots of references in the web).
- Worry about cache first, then go for –O flags, loop unrolling etc.

Last words

- Have fun with performance tweaking!
  Performance optimization is all about “cheating”.
- We will discuss your questions and progress next week.

- **Do not hesitate** to contact me when you’re stuck.
  The best coders became champions because they dared to ask lots of “stupid” questions in their lives.

Tobias Fuchs

tobias.fuchs@nm.ifi.lmu.de

www.mnm-team.org/~fuchst