Parallel Matrix Multiplication

Final Project

INF236: Parallel Programming

Kateřina Čížková, Luca Klingenberg

UNIVERSITY OF BERGEN

Table of contents

Introduction

- Complexity: $\mathcal{O}(n^3)$
- Consecutive memory access

Algorithm 1 matrix multiplication

Input: $\mathbf{A}, \mathbf{B}$
Output: $\mathbf{C}$ (the resulting matrix)

function MATMUL($\mathbf{A}, \mathbf{B}$)
    for $i = 0, \dots, n-1$ **do**
        for $j = 0, \dots, n-1$ **do**
            $c[i][j] = 0$
        for $k = 0, \dots, n-1$ **do**
            for $j = 0, \dots, n-1$ **do**
                $c[i][j] += a[i][k] \cdot b[k][j]$
    return $\mathbf{C}$

Table of contents

Strassen's Algorithm

- Blockwise matrix multiplication

$$
\begin{aligned}
\mathbf{A} \cdot \mathbf{B} &= \begin{pmatrix} \mathbf{A}_{00} & \mathbf{A}_{01} \\ \mathbf{A}_{10} & \mathbf{A}_{11} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{B}_{00} & \mathbf{B}_{01} \\ \mathbf{B}_{10} & \mathbf{B}_{11} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{A}_{00}\mathbf{B}_{00} + \mathbf{A}_{01}\mathbf{B}_{10} & \mathbf{A}_{00}\mathbf{B}_{01} + \mathbf{A}_{01}\mathbf{B}_{11} \\ \mathbf{A}_{10}\mathbf{B}_{00} + \mathbf{A}_{11}\mathbf{B}_{10} & \mathbf{A}_{10}\mathbf{B}_{01} + \mathbf{A}_{11}\mathbf{B}_{11} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{C}_{00} & \mathbf{C}_{01} \\ \mathbf{C}_{10} & \mathbf{C}_{11} \end{pmatrix} = \mathbf{C}
\end{aligned}
$$

- Idea: reduce number of multiplications from 8 to 7
- Complexity of Strassen's algorithm: $\mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2.8073})$
- Can be further improved by reusing intermediate results of additions and subtractions (Winograd's algorithm)
- Input matrices are zero-padded until the next power of two

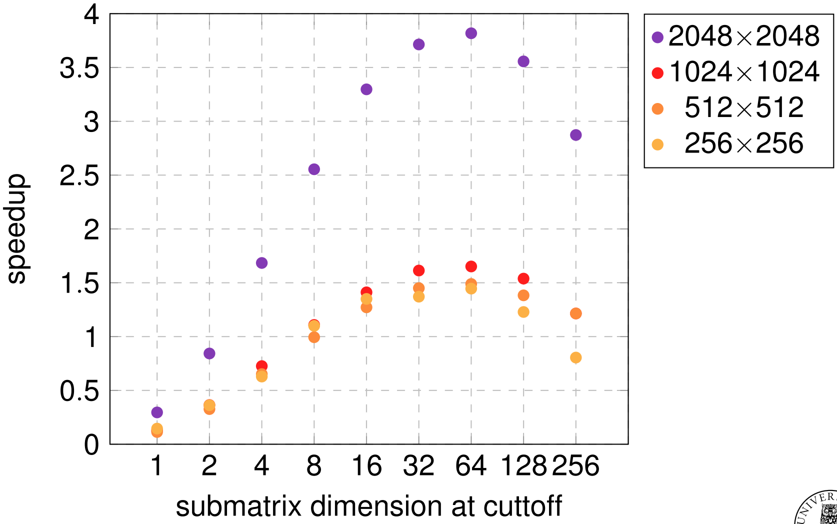
Strassen's Algorithm

Algorithm 2 Strassen's matrix multiplication

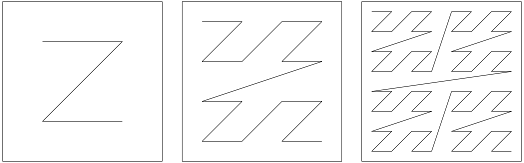
Input: $\mathbf{A}, \mathbf{B}$
Output: $\mathbf{C}$ (the resulting matrix)

function STRASSEN($\mathbf{A}, \mathbf{B}, n$)
    if n == cutoff **then**
        return MATMUL($\mathbf{A}, \mathbf{B}$)
    $\mathbf{P}_1 = \text{STRASSEN}(\mathbf{A}_{00} + \mathbf{A}_{11}, \mathbf{B}_{00} + \mathbf{B}_{11}, \frac{n}{2})$
    $\mathbf{P}_2 = \text{STRASSEN}(\mathbf{A}_{10} + \mathbf{A}_{11}, \mathbf{B}_{00}, \frac{n}{2})$
    $\mathbf{P}_3 = \text{STRASSEN}(\mathbf{A}_{00}, \mathbf{B}_{01} - \mathbf{B}_{11}, \frac{n}{2})$
    $\mathbf{P}_4 = \text{STRASSEN}(\mathbf{A}_{11}, \mathbf{B}_{10} - \mathbf{B}_{00}, \frac{n}{2})$
    $\mathbf{P}_5 = \text{STRASSEN}(\mathbf{A}_{00} + \mathbf{A}_{01}, \mathbf{B}_{11}, \frac{n}{2})$
    $\mathbf{P}_6 = \text{STRASSEN}(\mathbf{A}_{10} - \mathbf{A}_{00}, \mathbf{B}_{00} + \mathbf{B}_{01}, \frac{n}{2})$
    $\mathbf{P}_7 = \text{STRASSEN}(\mathbf{A}_{01} - \mathbf{A}_{11}, \mathbf{B}_{10} + \mathbf{B}_{11}, \frac{n}{2})$
    $\mathbf{C}_{00} = \mathbf{P}_1 + \mathbf{P}_4 - \mathbf{P}_5 + \mathbf{P}_7$
    $\mathbf{C}_{01} = \mathbf{P}_3 + \mathbf{P}_5$
    $\mathbf{C}_{10} = \mathbf{P}_2 + \mathbf{P}_4$
    $\mathbf{C}_{11} = \mathbf{P}_1 - \mathbf{P}_2 + \mathbf{P}_3 + \mathbf{P}_6$
    return C

Different values for the cutoff level



Z-ordering



$$\begin{array}{cccc} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{array}$$

1D array in memory:

$a_{00}\ a_{01}\ a_{02}\ a_{03}\ a_{10}\ a_{11}\ a_{12}\ \ldots$

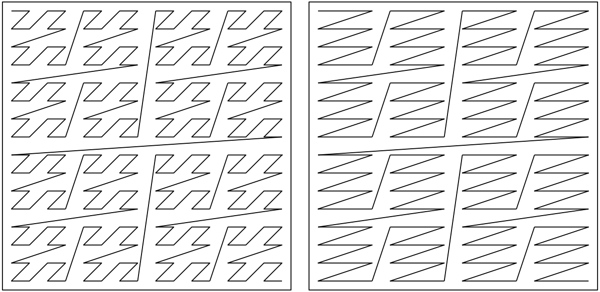
$$\begin{array}{cccc} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{array}$$

1D array in memory:

$a_{00}\ a_{01}\ a_{10}\ a_{11}\ a_{02}\ a_{03}\ a_{12}\ \ldots$

Z-ordering

- Matrices at cutoff level are stored row-wise



- Only pointers instead of the actual submatrices are passed to the recursive calls

Table of contents

Parallelization: Matrix multiplication

Algorithm 3 matrix multiplication

Input: $\mathbf{A}, \mathbf{B}$
Output: $\mathbf{C}$ (the resulting matrix)

#pragma omp parallel for
function MATMUL($\mathbf{A}, \mathbf{B}$)
    for $i = 0, \ldots, n-1$ **do**
        for $j = 0, \ldots, n-1$ **do**
            $c[i][j] = 0$
        for $k = 0, \ldots, n-1$ **do**
            for $j = 0, \ldots, n-1$ **do**
                $c[i][j] += a[i][k] \cdot b[k][j]$
    return C

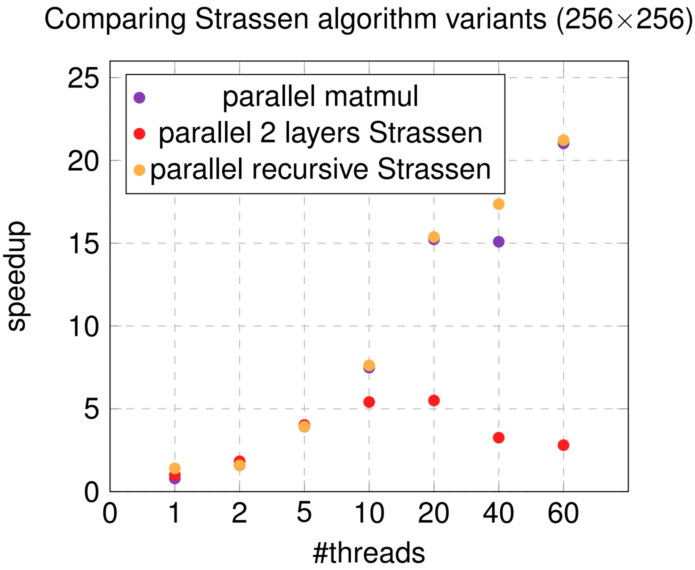
Parallelization: Strassen

- 2-layers variant
  - Use 2 hardcoded recursion levels of Strassen
  - Execute all additions and subtractions in parallel
  - At the cutoff level, execute all matrix multiplications in parallel

- Recursive variant
  - Use several recursive calls of Strassen
  - Execute all additions and subtractions in parallel
  - At the cutoff level, execute all matrix multiplications in parallel
  - Strassen is recursively applied until the submatrices have a dimension of 512×512

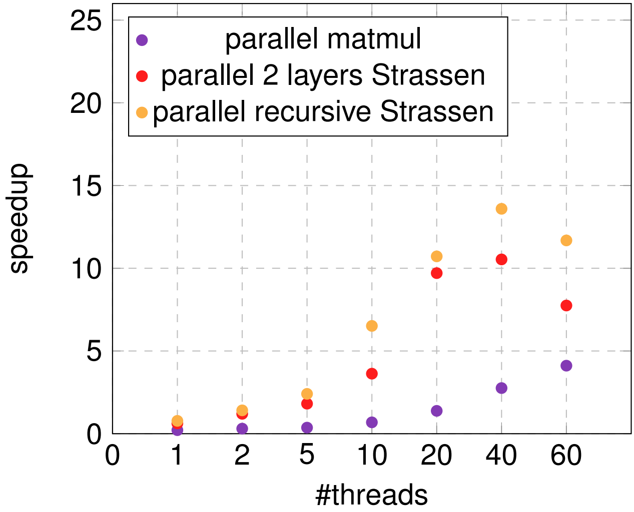
Table of contents

Experiments



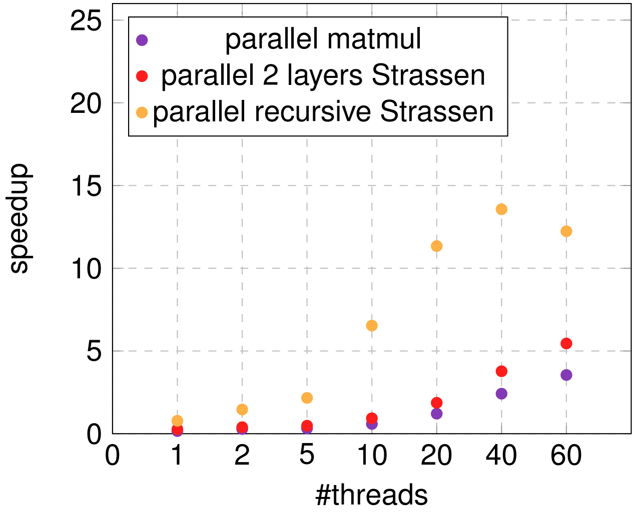
Experiments

Comparing Strassen algorithm variants (4096×4096)



Experiments

Comparing Strassen algorithm variants (8192×8192)



Experiments

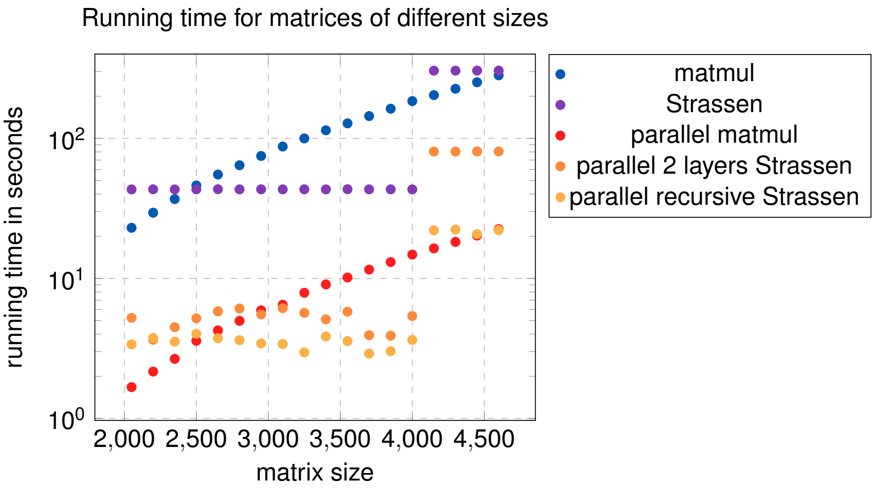


Table of contents

Conclusion

- With sufficient matrix size, the parallel variants of Strassen's algorithm perform much better
- For more than 40 threads, the parallel recursive Strassen version slows down again
- Zero padding could be improved