8. Hardware-Aware Numerics

Approaching supercomputing ...

8.1. Hardware-Awareness

Introduction

- Since numerical algorithms are ubiquitous, they have to run on a broad spectrum of processors or devices, resp.:
  - commodity CPU (Intel, AMD, . . .)
  - special supercomputing CPU (vector processors, . . .)
  - special-purpose processors such as GPU (NVIDIA, . . .) or the Cell Broadband Engine (in Sony's PlayStation)
  - other devices: PDA, iPhone, . . .
- While the classical concern of numerical algorithms lies on the algorithmic side (speed of convergence, complexity in terms of $O(N^k)$, accuracy in terms of $O(h^k)$, memory consumption), it has become obvious that this is not sufficient for performance, i. e. short run times – **implementational** aspects gain more and more in importance:
  - tailoring *data structures*
  - exploiting *pipelining*
  - exploiting *memory hierarchies* (the different cache levels, esp.)
  - exploiting *on-chip parallelism*

- Of course, there needs to be a balance between code performance on the one side and code portability on the other side:
  - **hardware-conscious**: increasing performance
  - **hardware-oblivious**: increasing performance by aligning algorithm design to general architectural features, without taking into account specific details of the respective architecture in the algorithm design
  - **hardware-aware**: comprises all measures that try to adapt algorithms to the underlying hardware, i.e. comprises hardware-conscious and hardware-oblivious

Relevance

- Program a matrix-vector or a matrix-matrix product of increasing dimension: at some point, performance will decrease tremendously.
- Staying two to four orders of magnitude below the processor's peak performance is not a rare event, if an algorithm is coded without additional considerations.
- One problem is the so-called **memory bottleneck** or **memory wall** – consider the average growth rates in the last years:
  - CPU performance: 60%
  - memory bandwidth: 23%
  - memory latency: 5%
- Another "hot topic" arises from today's ubiquitous parallelism in present multi-core and upcoming many-core systems. Take a moment to think about possible parallelization strategies for the Jacobi or the Gauß-Seidel methods discussed in the chapter on iterative schemes.
- Tackling such problems is one focus of *Scientific Computing*.
- In this chapter, we will concentrate on one aspect: increasing cache-efficiency for matrix-matrix multiplication.

8.2. Space-Filling Curves

Introduction

- An unconventional strategy for cache-efficiency
- Origin of the idea: analysis and topology ("topological monsters")
- Nice example of a construct from pure mathematics that gets practical relevance decades later
- Definition of a **space-filling curve (SFC)**, for reasons of simplicity only in 2 D:
  - Curve: image of a continuous mapping of the unit interval $[0, 1]$ onto the unit square $[0, 1]^2$
  - Space-filling: curve covers the whole unit square (mapping is surjective) and, hence, covers an area greater than zero(!)

$$f: \ [0,1] =: I \quad \rightarrow \quad Q := [0,1]^2, \quad f \text{ surjective and continuous}$$

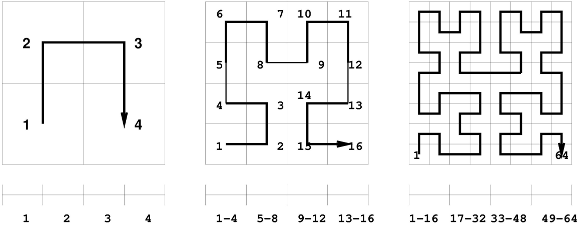
- Prominent representatives:
  - *Hilbert's curve*: 1891, the most famous space-filling curve
  - *Peano's curve*: 1890, oldest space-filling curve
  - *Lebesgue's curve*: quadtree principle, probably the most important SFC for computer science

Hilbert's SFC

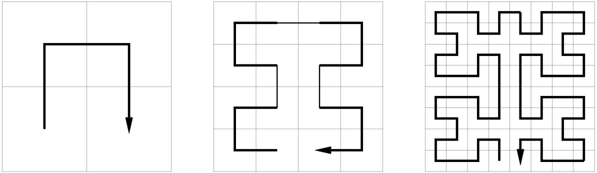
- The construction follows the geometric conception: if I can be mapped onto Q in the space-filling sense, then each of the four congruent subintervals of I can be mapped to one of the four quadrants of Q in the space-filling sense, too.
- Recursive application of this partitioning and allocation process preserving
  - *Neighborhood relations*: neighboring subintervals in I are mapped onto neighboring subsquares of Q.
  - *Subset relations (inclusion)*: from $I_1 \subseteq I_2$ follows $f(I_1) \subseteq f(I_2)$
- Limit case: Hilbert's curve
  - From the correspondence of nestings of intervals in I and nestings of squares in Q, we get pairs of points in I and of corresponding image points in Q.
  - Of course, the iterative steps in this generation process are of practical relevance, not the limit case (the SFC) itself.
    - Start with a **generator** (defines the order in which the subsquares are "visited")
    - Apply generator in each subsquare (with appropriate similarity transformations)
    - Connect the open ends

Generation Processes with Hilbert's Generator

- Classical version of Hilbert:



- Variant of Moore:

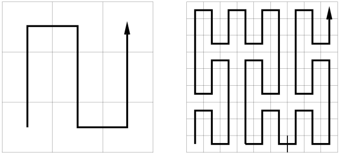


- Modulo symmetry, these are the only two possibilities!

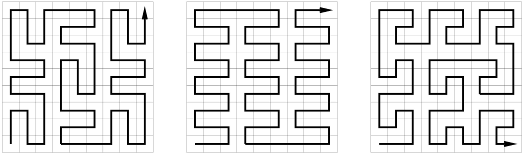
Peano's SFC

- Ancestor of all SFCs
- Subdivision of I and Q into nine congruent subdomains
- Definition of a leitmotiv, again, defines the order of visit

3	4	9
2	5	8
1	6	7



- Now, there are 273 different (modulo symmetry) possibilities to recursively apply the generator preserving neighborhood and inclusion



Serpentine type (left and center) and meander type (right)

8.3. Matrix-Matrix Multiplication

Relevance and Standard Algorithm

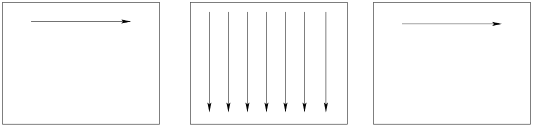
- Matrix-matrix multiplication is not a such frequently used building block of numerical algorithms as matrix-vector multiplication is.
- Nevertheless several appearances:
  - Computational chemistry: computing changes of state in chemical systems
  - Signal processing: performing some classes of transforms
- Standard sequential algorithm for two quadratic matrices $A, B \in \mathbb{R}^{M,M}$:

```
for i=1 to n do
    for j=1 to n do
        c[i,j] := 0;
        for k=1 to n do
        c[i,j] := c[i,j]+a[i,k]*b[k,j];
```

- That is: a sequence of M^2 scalar products of two vectors of length M
- For full matrices we get cubic complexity.

Observation

- In a single iteration of the outer loop indexed by i, row i of matrix A and all rows of matrix B are read, while row i of matrix C is written.



- Consequence: once M reaches a certain size, B won't fit completely into the cache any more, and performance will fall dramatically (frequent cache misses and, hence, main memory accesses during each outer iteration step, i. e. row of A)
- Remedy: a recursive variant working with blocks of B only instead of the whole matrix B

Recursive Block-Oriented Algorithm

- Subdivide both A and B into four smaller submatrices of consistent dimensions:

$$A = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \quad B = \begin{pmatrix} B_{00} & B_{01} \\ B_{10} & B_{11} \end{pmatrix}$$

- The matrix product then reads

$$C = \begin{pmatrix} A_{00}B_{00} + A_{01}B_{10} & A_{00}B_{01} + A_{01}B_{11} \\ A_{10}B_{00} + A_{11}B_{10} & A_{10}B_{01} + A_{11}B_{11} \end{pmatrix}$$

  (compare the product of two 2×2-matrices)

- If the blocks of B are still too large for the cache, this subdivision step can be applied recursively to finally overcome the cache problem.
- Today, block-recursive approaches are widespread techniques which, by construction, leads to inherently good data access patterns and, thus, to good cache performance.
- This strategy is also important for parallel matrix-matrix algorithms.

8.4. Peano-Based Matrix-Matrix Product

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$\boxed{1\ 1\ 1} \rightarrow$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} \mathbf{1} & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \boxed{3} & 4 & 9 \end{pmatrix} \begin{pmatrix} \boxed{1} & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \boxed{3} & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \longrightarrow \boxed{2\ 1\ 2} \longrightarrow \boxed{3\ 1\ 3} \longrightarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \boxed{4} & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ \boxed{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \boxed{3} & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow \boxed{5\ 2\ 2} \rightarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & \mathbf{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} \mathbf{1} & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \to \boxed{2\ 1\ 2} \to \boxed{3\ 1\ 3} \to \boxed{4\ 2\ 3} \to \boxed{5\ 2\ 2} \to \boxed{6\ 2\ 1} \to$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow \boxed{5\ 2\ 2} \rightarrow \boxed{6\ 2\ 1} \rightarrow \boxed{7\ 3\ 1} \downarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \mathbf{3} & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\;1\;1} \rightarrow \boxed{2\;1\;2} \rightarrow \boxed{3\;1\;3} \rightarrow \boxed{4\;2\;3} \rightarrow \boxed{5\;2\;2} \rightarrow \boxed{6\;2\;1} \rightarrow \boxed{7\;3\;1} \downarrow$$

$$\leftarrow \boxed{8\;3\;2}$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \mathbf{3} & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \mathbf{3} & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow \boxed{5\ 2\ 2} \rightarrow \boxed{6\ 2\ 1} \rightarrow \boxed{7\ 3\ 1} \rightarrow \boxed{8\ 3\ 2} \rightarrow \boxed{9\ 3\ 3} \rightarrow$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$
\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix}
\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \mathbf{4} & 9 \end{pmatrix}
=
\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \mathbf{4} & 9 \end{pmatrix}
$$

$$
\begin{tikzcd}
\boxed{1\;1\;1} \arrow[r] & \boxed{2\;1\;2} \arrow[r] & \boxed{3\;1\;3} \arrow[r] & \boxed{4\;2\;3} \arrow[r] & \boxed{5\;2\;2} \arrow[r] & \boxed{6\;2\;1} \arrow[r] & \boxed{7\;3\;1} \arrow[d] \\
 & & & {} & \boxed{9\;4\;4} \arrow[l] & \boxed{9\;3\;3} \arrow[l] & \boxed{8\;3\;2} \arrow[l]
\end{tikzcd}
$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \boxed{\mathbf{8}} \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \boxed{\mathbf{4}} & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & \boxed{\mathbf{5}} & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \to \boxed{2\ 1\ 2} \to \boxed{3\ 1\ 3} \to \boxed{4\ 2\ 3} \to \boxed{5\ 2\ 2} \to \boxed{6\ 2\ 1} \to \boxed{7\ 3\ 1}$$

$$\leftarrow \boxed{8\ 4\ 5} \leftarrow \boxed{9\ 4\ 4} \leftarrow \boxed{9\ 3\ 3} \leftarrow \boxed{8\ 3\ 2}$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \mathbf{4} & 9 \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\,1\,1} \to \boxed{2\,1\,2} \to \boxed{3\,1\,3} \to \boxed{4\,2\,3} \to \boxed{5\,2\,2} \to \boxed{6\,2\,1} \to \boxed{7\,3\,1}$$

$$\to \boxed{8\,3\,2} \to \boxed{9\,3\,3} \to \boxed{9\,4\,4} \to \boxed{8\,4\,5} \to \boxed{7\,4\,6} \to$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & \boxed{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & \boxed{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & \boxed{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow \boxed{5\ 2\ 2} \rightarrow \boxed{6\ 2\ 1} \rightarrow \boxed{7\ 3\ 1}$$

$$\downarrow$$

$$\leftarrow \boxed{6\ 5\ 6} \leftarrow \boxed{7\ 4\ 6} \leftarrow \boxed{8\ 4\ 5} \leftarrow \boxed{9\ 4\ 4} \leftarrow \boxed{9\ 3\ 3} \leftarrow \boxed{8\ 3\ 2}$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 →

8 3 2 → 9 3 3 → 9 4 4 → 8 4 5 → 7 4 6 → 6 5 6 → 5 5 5 →

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \mathbf{4} & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \mathbf{4} & 9 \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 → 8 3 2 → 9 3 3 → 9 4 4 → 8 4 5 → 7 4 6 → 6 5 6 → 5 5 5 → 4 5 4 →

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \boxed{\mathbf{3}} & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & \boxed{\mathbf{6}} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \boxed{\mathbf{4}} & 9 \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 ↓

5 5 5 ← 6 5 6 ← 7 4 6 ← 8 4 5 ← 9 4 4 ← 9 3 3 ← 8 3 2

↓ 4 5 4 → 3 6 4 →

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 ↓

5 5 5 ← 6 5 6 ← 7 4 6 ← 8 4 5 ← 9 4 4 ← 9 3 3 ← 8 3 2

↓ 4 5 4 → 3 6 4 → 2 6 5 →

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} \mathbf{1} & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & \mathbf{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$$\boxed{1\,1\,1} \to \boxed{2\,1\,2} \to \boxed{3\,1\,3} \to \boxed{4\,2\,3} \to \boxed{5\,2\,2} \to \boxed{6\,2\,1} \to \boxed{7\,3\,1}$$

$$\boxed{5\,5\,5} \leftarrow \boxed{6\,5\,6} \leftarrow \boxed{7\,4\,6} \leftarrow \boxed{8\,4\,5} \leftarrow \boxed{9\,4\,4} \leftarrow \boxed{9\,3\,3} \leftarrow \boxed{8\,3\,2}$$

$$\boxed{4\,5\,4} \to \boxed{3\,6\,4} \to \boxed{2\,6\,5} \to \boxed{1\,6\,6} \to$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} \mathbf{1} & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$

$\boxed{1\ 1\ 1} \rightarrow \boxed{2\ 1\ 2} \rightarrow \boxed{3\ 1\ 3} \rightarrow \boxed{4\ 2\ 3} \rightarrow \boxed{5\ 2\ 2} \rightarrow \boxed{6\ 2\ 1} \rightarrow \boxed{7\ 3\ 1}$

$\downarrow$

$\boxed{5\ 5\ 5} \leftarrow \boxed{6\ 5\ 6} \leftarrow \boxed{7\ 4\ 6} \leftarrow \boxed{8\ 4\ 5} \leftarrow \boxed{9\ 4\ 4} \leftarrow \boxed{9\ 3\ 3} \leftarrow \boxed{8\ 3\ 2}$

$\downarrow$

$\boxed{4\ 5\ 4} \rightarrow \boxed{3\ 6\ 4} \rightarrow \boxed{2\ 6\ 5} \rightarrow \boxed{1\ 6\ 6} \rightarrow \boxed{1\ 7\ 7} \rightarrow$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ \mathbf{2} & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix}$$

$$\begin{tikzcd}
\boxed{1\,1\,1} \arrow[r] & \boxed{2\,1\,2} \arrow[r] & \boxed{3\,1\,3} \arrow[r] & \boxed{4\,2\,3} \arrow[r] & \boxed{5\,2\,2} \arrow[r] & \boxed{6\,2\,1} \arrow[r] & \boxed{7\,3\,1} \arrow[d] \\
\boxed{5\,5\,5} \arrow[d] & \boxed{6\,5\,6} \arrow[l] & \boxed{7\,4\,6} \arrow[l] & \boxed{8\,4\,5} \arrow[l] & \boxed{9\,4\,4} \arrow[l] & \boxed{9\,3\,3} \arrow[l] & \boxed{8\,3\,2} \arrow[l] \\
\boxed{4\,5\,4} \arrow[r] & \boxed{3\,6\,4} \arrow[r] & \boxed{2\,6\,5} \arrow[r] & \boxed{1\,6\,6} \arrow[r] & \boxed{1\,7\,7} \arrow[r] & \boxed{2\,7\,8} \arrow[r] & {}
\end{tikzcd}$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ \mathbf{3} & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix}$$

$$\begin{tikzcd}
\boxed{1\,1\,1} \arrow[r] & \boxed{2\,1\,2} \arrow[r] & \boxed{3\,1\,3} \arrow[r] & \boxed{4\,2\,3} \arrow[r] & \boxed{5\,2\,2} \arrow[r] & \boxed{6\,2\,1} \arrow[r] & \boxed{7\,3\,1} \arrow[d] \\
\boxed{5\,5\,5} \arrow[d] & \boxed{6\,5\,6} \arrow[l] & \boxed{7\,4\,6} \arrow[l] & \boxed{8\,4\,5} \arrow[l] & \boxed{9\,4\,4} \arrow[l] & \boxed{9\,3\,3} \arrow[l] & \boxed{8\,3\,2} \arrow[l] \\
\boxed{4\,5\,4} \arrow[r] & \boxed{3\,6\,4} \arrow[r] & \boxed{2\,6\,5} \arrow[r] & \boxed{1\,6\,6} \arrow[r] & \boxed{1\,7\,7} \arrow[r] & \boxed{2\,7\,8} \arrow[r] & \boxed{3\,7\,9} \arrow[d] \\
 & & & & & & {}
\end{tikzcd}$$

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

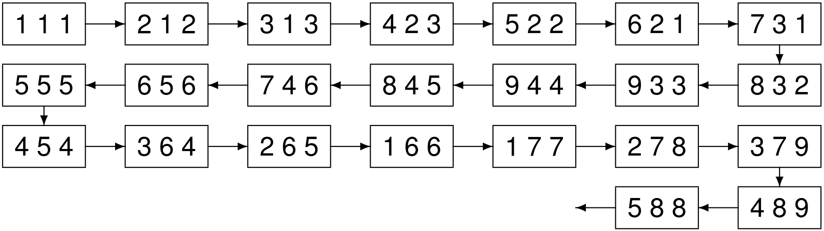
$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & \boxed{\mathbf{4}} & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \boxed{\mathbf{8}} \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \boxed{\mathbf{9}} \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1
↓
8 3 2 → 9 3 3 → 9 4 4 → 8 4 5 → 7 4 6 → 6 5 6 → 5 5 5
↓
4 5 4 → 3 6 4 → 2 6 5 → 1 6 6 → 1 7 7 → 2 7 8 → 3 7 9
↓
4 8 9 →

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

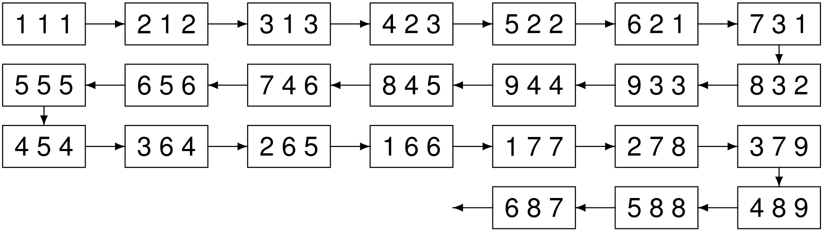
$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & \mathbf{5} & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix}$$



Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

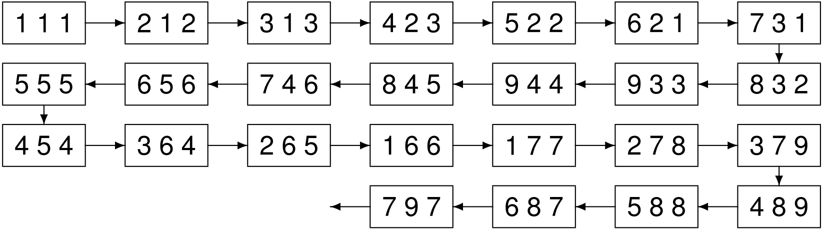
$$\begin{pmatrix} 1 & \mathbf{6} & 7 \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 6 & \mathbf{7} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$



Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & \boxed{\mathbf{7}} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \boxed{\mathbf{9}} \end{pmatrix} = \begin{pmatrix} 1 & 6 & \boxed{\mathbf{7}} \\ 2 & 5 & 8 \\ 3 & 4 & 9 \end{pmatrix}$$



Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & \mathbf{8} \\ 3 & 4 & 9 \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 ↓

5 5 5 ← 6 5 6 ← 7 4 6 ← 8 4 5 ← 9 4 4 ← 9 3 3 ← 8 3 2

↓ 4 5 4 → 3 6 4 → 2 6 5 → 1 6 6 → 1 7 7 → 2 7 8 → 3 7 9 ↓

← 8 9 8 ← 7 9 7 ← 6 8 7 ← 5 8 8 ← 4 8 9

Start: Multiplication of 3×3 Matrices

An optimal (Peano-)order of execution:

$$\begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix} \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix} = \begin{pmatrix} 1 & 6 & 7 \\ 2 & 5 & 8 \\ 3 & 4 & \mathbf{9} \end{pmatrix}$$

1 1 1 → 2 1 2 → 3 1 3 → 4 2 3 → 5 2 2 → 6 2 1 → 7 3 1 ↓

5 5 5 ← 6 5 6 ← 7 4 6 ← 8 4 5 ← 9 4 4 ← 9 3 3 ← 8 3 2

↓

4 5 4 → 3 6 4 → 2 6 5 → 1 6 6 → 1 7 7 → 2 7 8 → 3 7 9 ↓

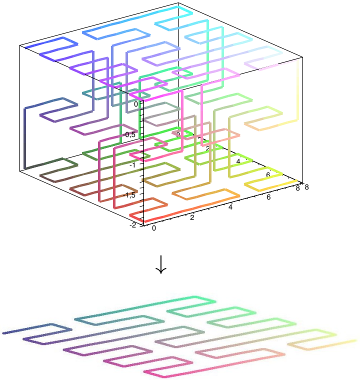
9 9 9 ← 8 9 8 ← 7 9 7 ← 6 8 7 ← 5 8 8 ← 4 8 9

Matrix Multiplication and 3D/2D Peano Traversals

- *inherently cache efficient* 3D-traversal of the block operations `C[i,j] += A[i,k] * B[k,j]` using a Peano curve
- projections of 3D curve to 2D-planes lead to 2D Peano curves
- 2D-planes correspond to the indices of A, B, and C: (i, k), (k, j), and (i, j)

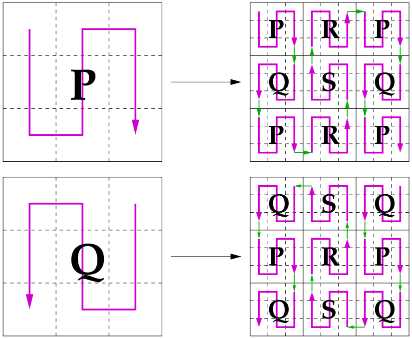
$\Rightarrow$ use Peano layout for matrices

Goal: strictly local element access



Block-Recursive Peano Element Order

- indexing according to iteration of a Peano curve



- stopped on *L1 blocks*
  (size tuned to L1 cache → 2 matrix blocks should fit)
- use column-major layout within L1-blocks

Block-Recursive Multiplication

- multiplication of block matrices (cmp. 3×3-scheme):

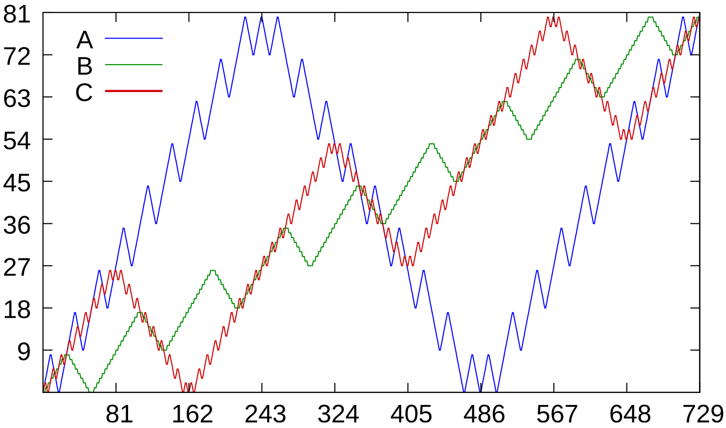
$$\begin{pmatrix} P_{A0} & R_{A5} & P_{A6} \\ Q_{A1} & S_{A4} & Q_{A7} \\ P_{A2} & R_{A3} & P_{A8} \end{pmatrix} \begin{pmatrix} P_{B0} & R_{B5} & P_{B6} \\ Q_{B1} & S_{B4} & Q_{B7} \\ P_{B2} & R_{B3} & P_{B8} \end{pmatrix} = \begin{pmatrix} P_{C0} & R_{C5} & P_{C6} \\ Q_{C1} & S_{C4} & Q_{C7} \\ P_{C2} & R_{C3} & P_{C8} \end{pmatrix}$$

- leads to **eight** different combinations (w.r.t. block numbering):

$$\begin{array}{cccc} PP \to P & QR \to S & RS \to R & SQ \to Q \\ PR \to R & QP \to Q & RQ \to P & SS \to S \end{array}$$

- all schemes similar to $PP \to P$ (but reverse order)

Access Pattern to the Matrix Blocks

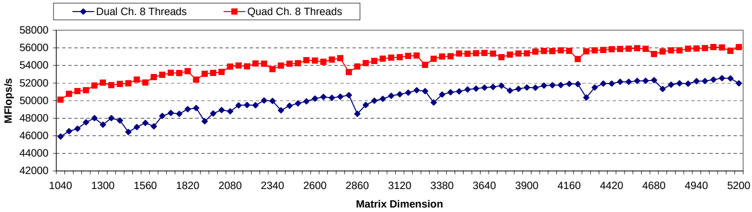


- Increment/Decrement access to elements
- $\mathcal{O}(k^3)$ operations on any block of k^2 elements

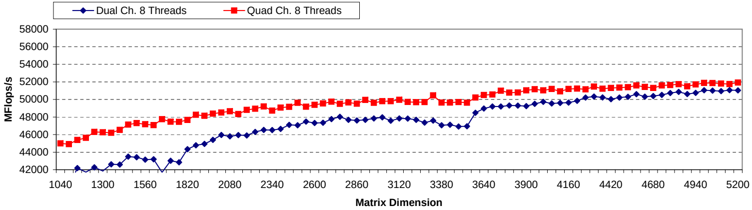
Performance – Memory Latency & Bandwidth

"Dual" vs. "Quad" channel memory (Xeon, 2×quadcore)

TifaMMy:



GotoBLAS:



Towards LU and ILU Decomposition

1. Sparse-Dense Matrix Operations:

- extension of the Peano approach to sparse matrices
- tree-oriented storage of dense and sparse matrices
  (L1 blocks zero, dense, or sparse)

2. Parallel LU Decomposition:

- block-oriented LU decomposition based on Peano curve
- shared-Memory parallelisation with OpenMP 3

3. Towards ILU Decomposition:

- is it feasible at all?
- increase level of parallelism during parallelisation

Extension for Sparse Matrices

Data structure:

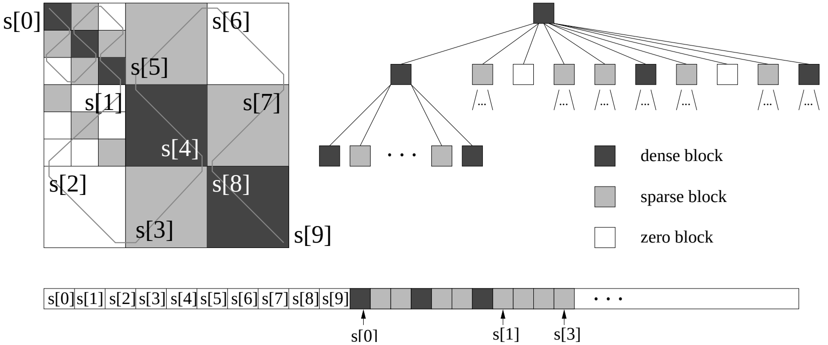
1. allow zero and dense blocks as L1 blocks
2. allow *compressed sparse row* (CSR) blocks as L1 blocks
3. adopt quadtree-like storage for matrices

Algorithm:

1. keep block-recursive Peano scheme
2. skip operations involving zero blocks
3. implement L1 operations on dense and sparse blocks

"Sparse Peano Tree" to Store Sparse Matrices

Tree-structure sequentialised in Peano order:



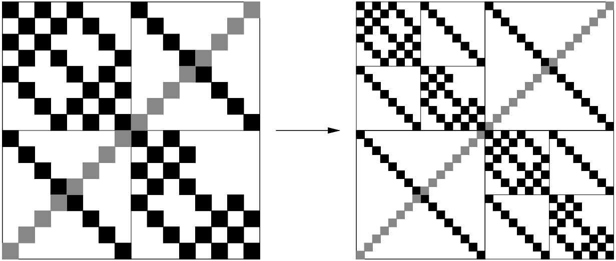
→ modified depth-first traversal (child information in parent node)

Matrix Exponentials in Quantum Control

- quantum states modelled by evolution matrices:

$$U^{(r)}(t_k) \;=\; e^{-i\Delta t H_k^{(r)}} e^{-i\Delta t H_{k-1}^{(r)}} \cdots e^{-i\Delta t H_1^{(r)}}$$

- wanted: exponential function of sparse matrices H_k:



- computed via Chebyshev polynomials
  $\rightarrow$ requires sparse-dense matrix multiplication