# Parallelization of Multiscale-Based Grid Adaptation Using Space Filling Curves

**Silvia-Sorana Mogosan**

**Paris, 22.01.2009**

# Outline

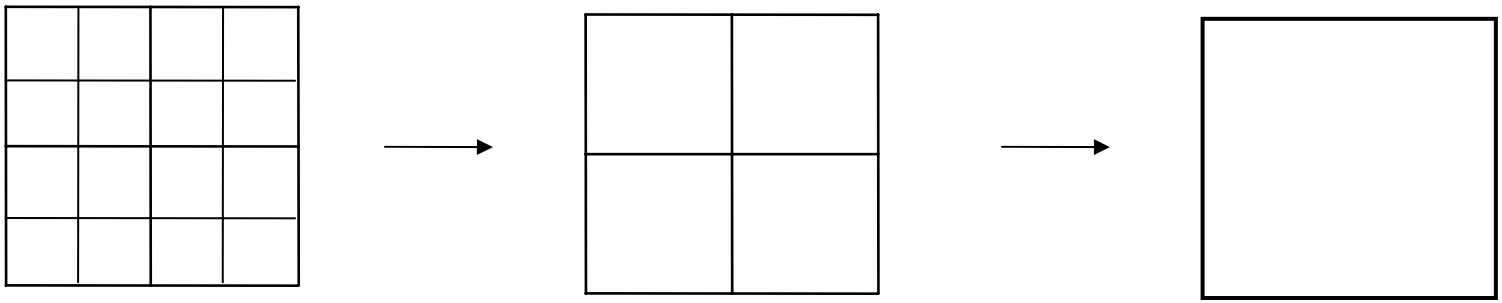
- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Outline

- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Multiscale Setting

- **Starting point**: a hierarchy of nested grids provided with cell averages




- **Goal**: to compress data
- The coarse cells averages can be computed starting on the finest level

$$\hat{u}_L \longleftrightarrow \hat{u}_{L-1} \longleftrightarrow \dots \longleftrightarrow \hat{u}_1 \longleftrightarrow \hat{u}_0$$
$$\qquad\qquad\quad d_{L-1} \qquad\quad \dots \qquad\quad d_1 \qquad\quad d_0$$

- Information is destroyed by averaging process, we store these information by means of details

# Multiscale Analysis

- Encoding

. The coarse grid average – a linear combination of the corresponding fine grid averages

$$\hat{u}_{l,\mathbf{k}} = \sum_{\mathbf{r}\in M^0_{l,\mathbf{k}}} \frac{|V_{l+1,\mathbf{r}}|}{|V_{l,\mathbf{k}}|}\hat{u}_{l+1,\mathbf{r}} =: \sum_{\mathbf{r}\in M^0_{l,\mathbf{k}}} m^{l,0}_{\mathbf{r},\mathbf{k}}\hat{u}_{l+1,\mathbf{r}}$$

. The update between 2 successive refinement levels is stored by details

$$d_{l,\mathbf{k},\mathbf{e}} = \sum_{\mathbf{r}\in M^{\mathbf{e}}_{l,\mathbf{k}}\subset I_l} m^{l,\mathbf{e}}_{\mathbf{r},\mathbf{k}}\hat{u}_{l+1,\mathbf{r}}$$

- Decoding

$$\hat{u}_{l+1,\mathbf{k}} = \sum_{\mathbf{r}\in G^0_{l,\mathbf{k}}\subset I_l} g^{l,0}_{\mathbf{r},\mathbf{k}}\hat{u}_{l,\mathbf{r}} + \sum_{\mathbf{e}\in E^*}\sum_{\mathbf{r}\in G^{\mathbf{e}}_{l,\mathbf{k}}\subset I_l} g^{l,\mathbf{e}}_{\mathbf{r},\mathbf{k}} d_{l,\mathbf{r},\mathbf{e}}$$

# Multiscale Grid Adaptation

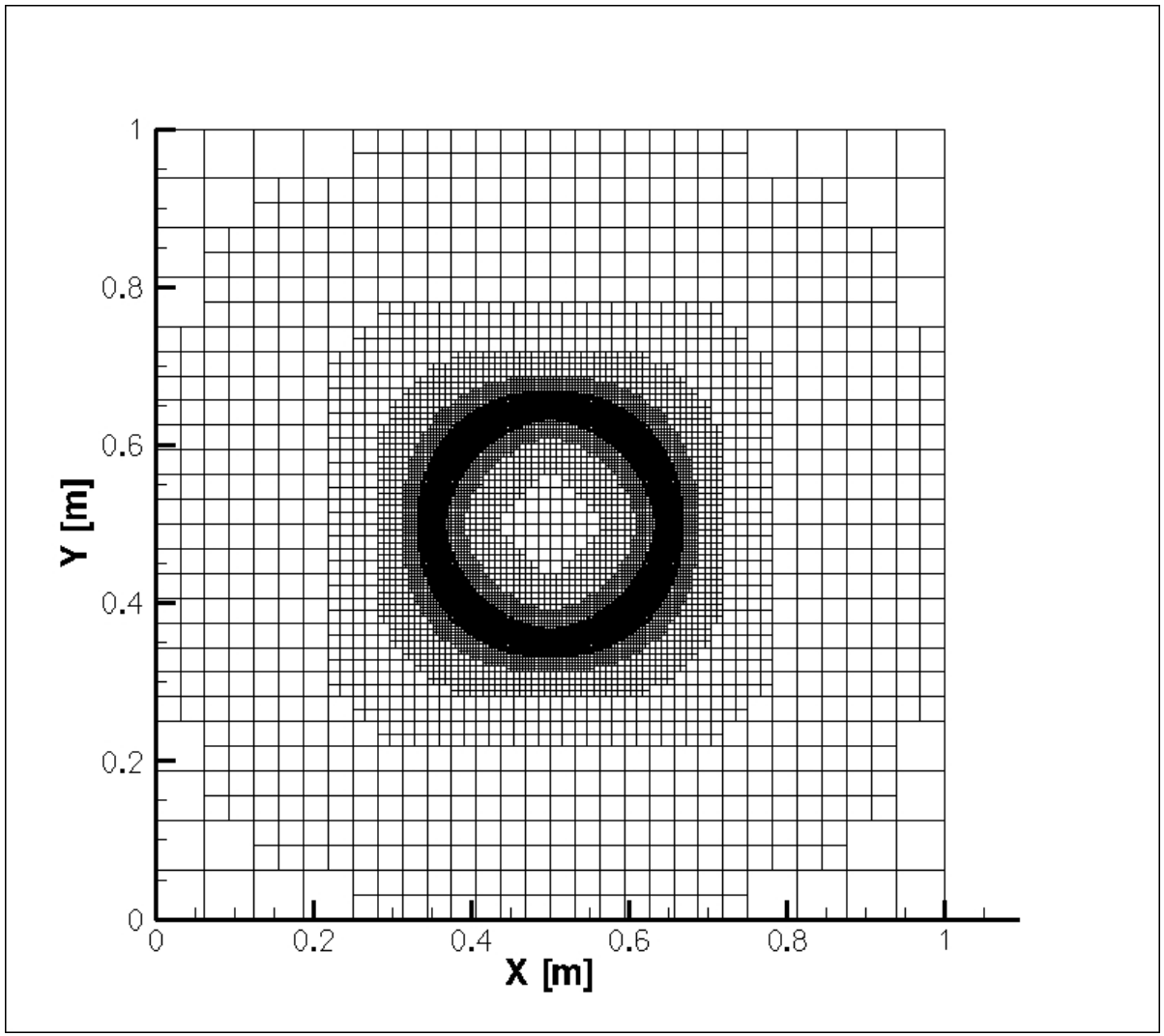
- Multiscale transformation (fine -> coarse)
- Thresholding (discard non-significant details)
- Prediction (predict which details could become significant at the next time step)
- Grading (the grid should have the structure of a graded tree)
- Inverse multiscale transformation (coarse ->fine)

# Outline

- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Partitioning Scheme

Starting point of the partitioning scheme: locally refined grid




L – finest level
0 – coarsest level

For each cell:
- the level *l*
- the multiindex *(i,j)*
  $i=0, \ldots, 2^l-1$
  $j=0, \ldots, 2^l-1$
- the cell average

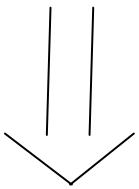
⇓

The key : *(l, (i,j))*

# Partitioning Scheme

## Goals:

- load-balancing
- little communication between processors
- partitions should be determined at runtime



## Space Filling Curves

- cheap
- help to simplify the implementation of the parallel algorithm

# Space Filling Curve

## Definition:

$$f \; : \; [0,1] =: I \; \mapsto \Omega \subset \mathbb{R}^d$$

*f* continuous and surjective

## Utility:

- A SFC can also be used for the inverse mapping *f* from $[0,1]^d$, d=2,3, to the unit interval *I*

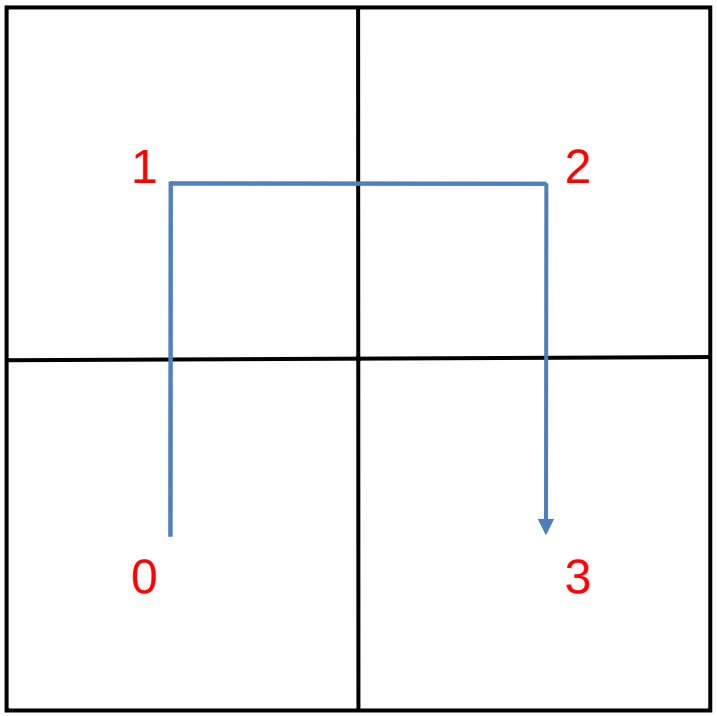
- We cut the interval *I* into disjoint sub-intervals $I_j$ with

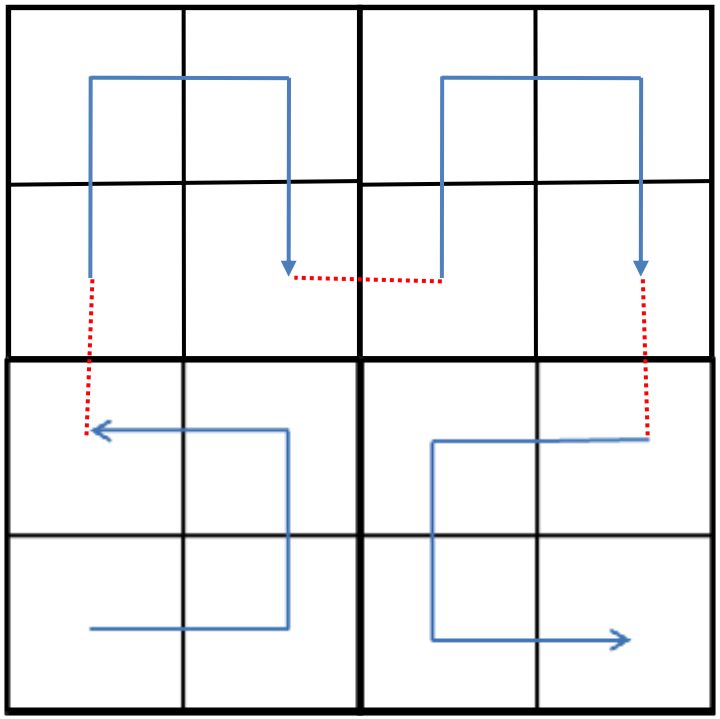
$$\bigcup_j I_j = I \qquad \bigcup_j f(I_j) \supset \Omega$$

- Perfect load balance and small separators between partitions

- However, the boundary of the geometrical sets $\partial f(I_j) \setminus \partial\Omega$ is larger than the optimal separators in general

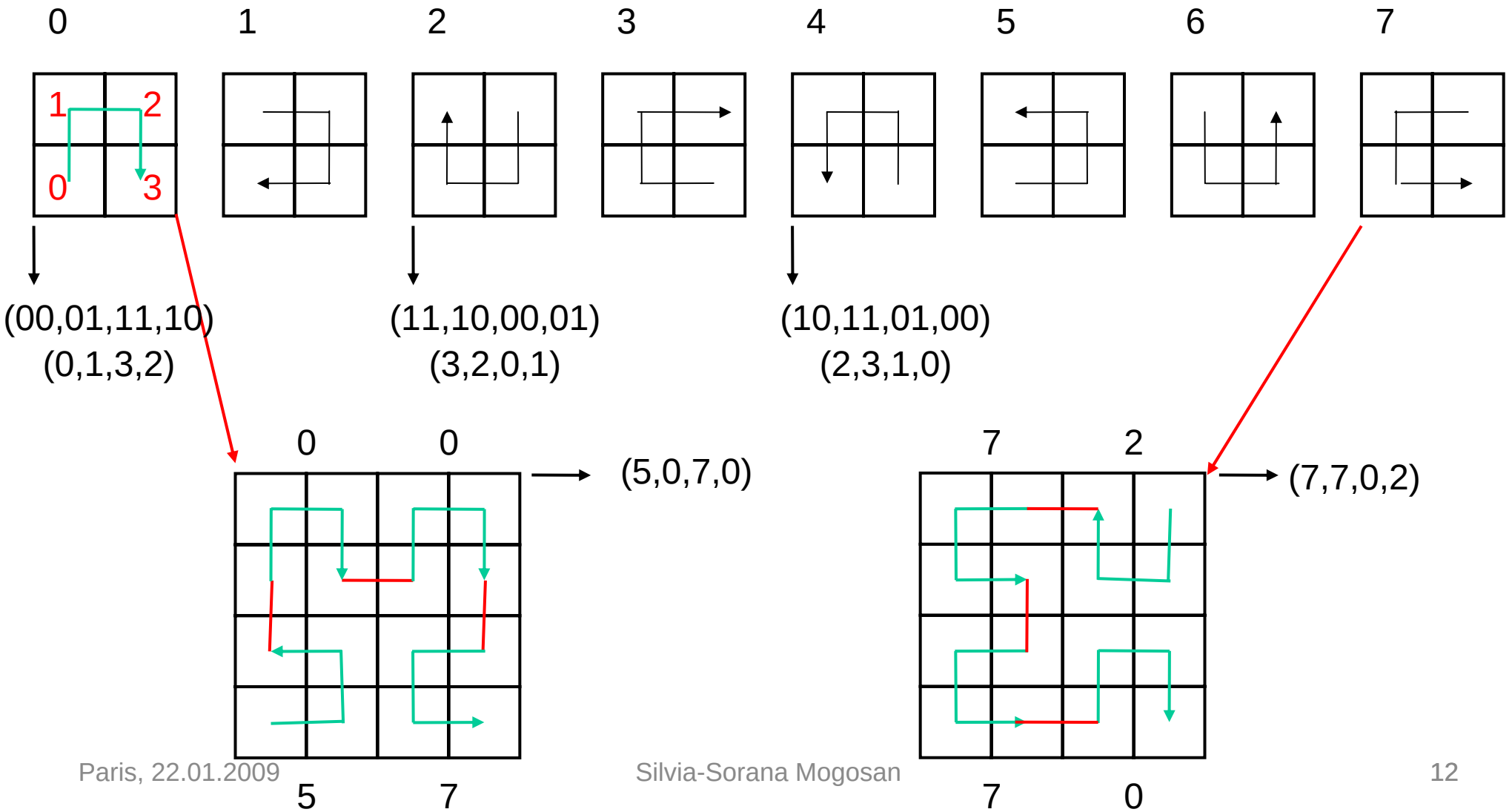
# Hilbert SFC



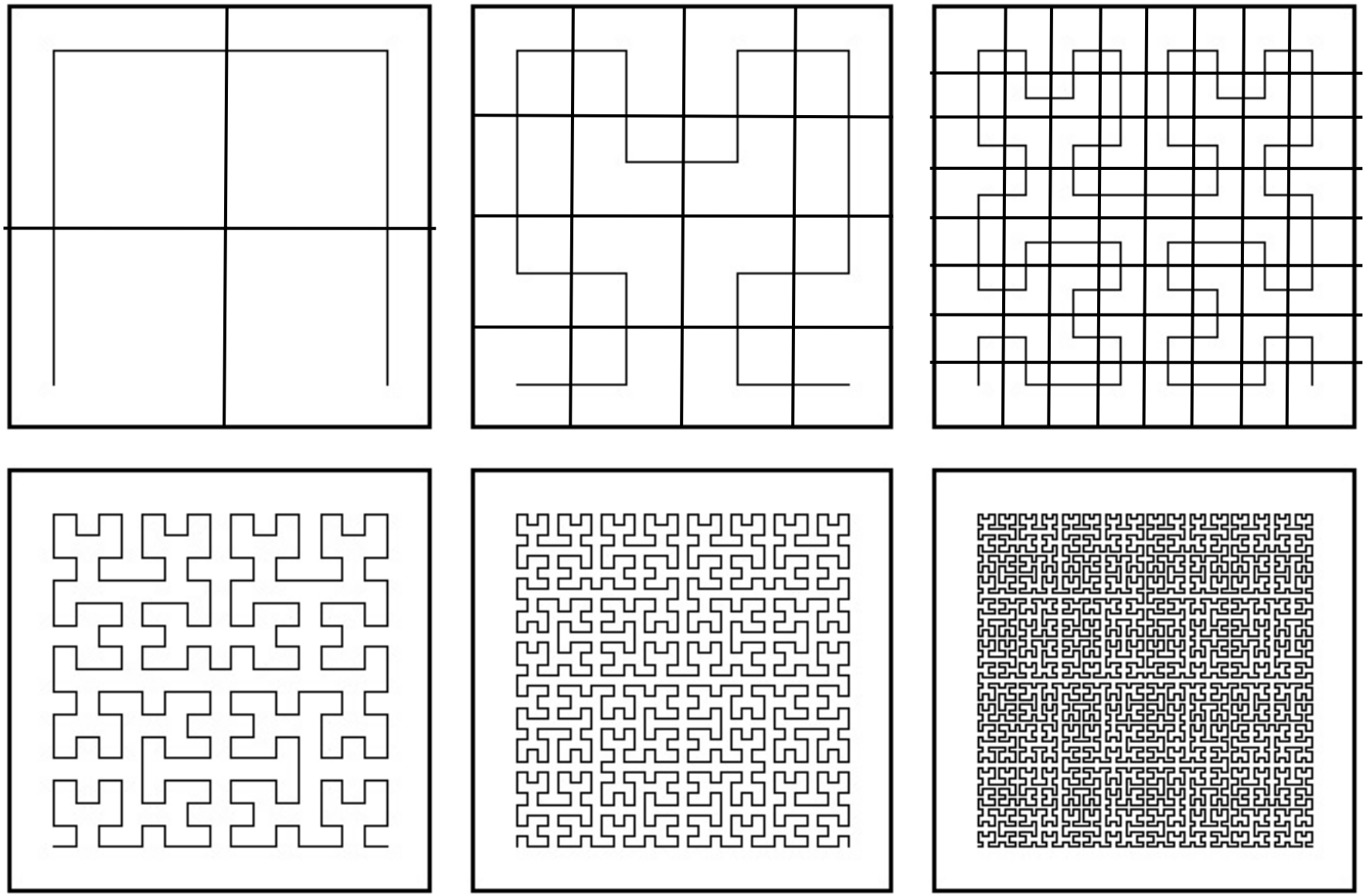



# 2D Hilbert SFC

## SFC 2D templates

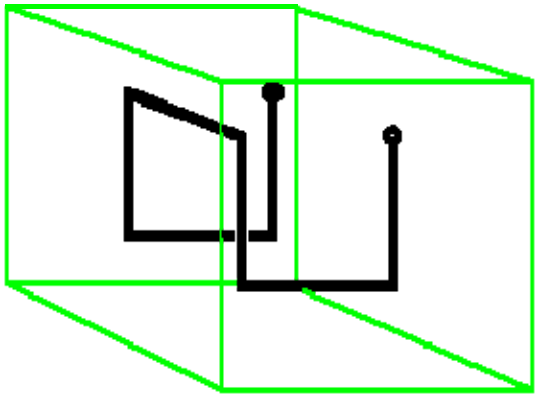


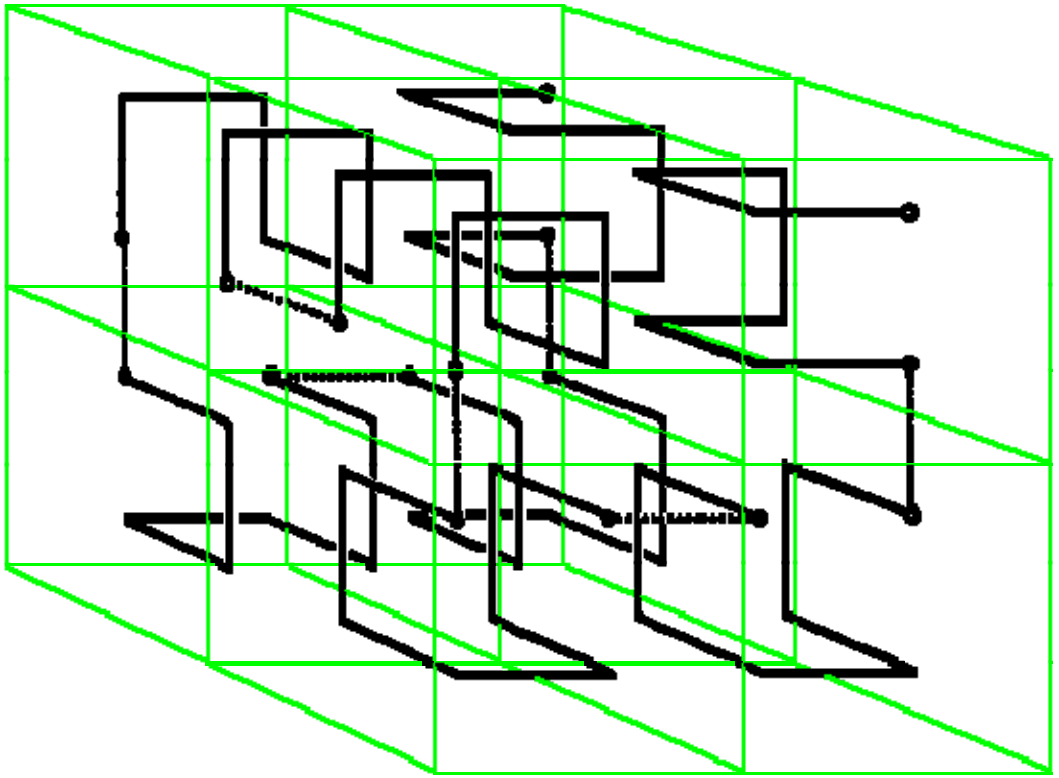
# 2D Hilbert SFC



2D Hilbert curve – first 6 refinement steps

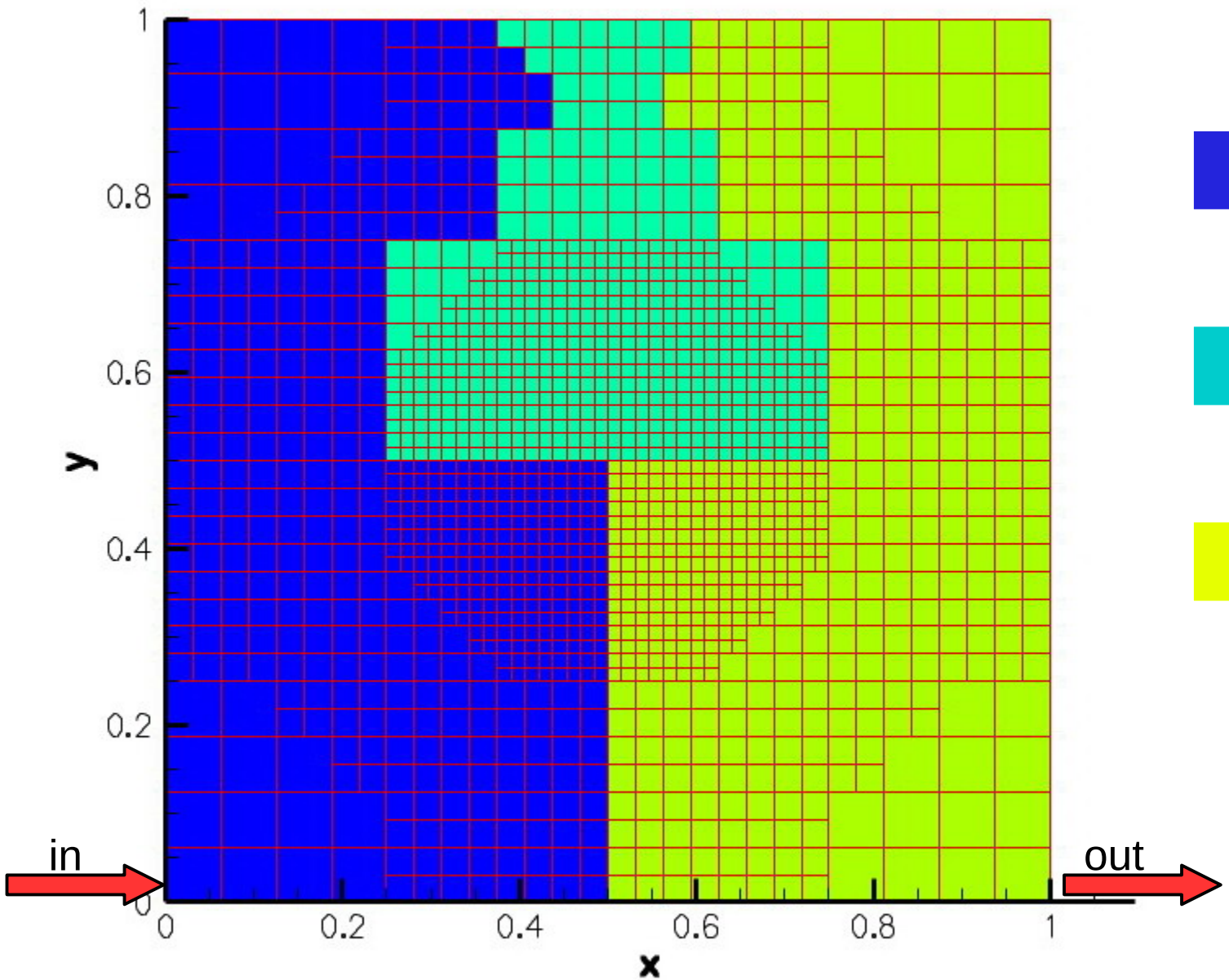
# 3D Hilbert SFC





3D Hilbert curve – first 2 refinement steps

# Data Distribution




Data on processor 0

Data on processor 1

Data on processor 2

# Outline

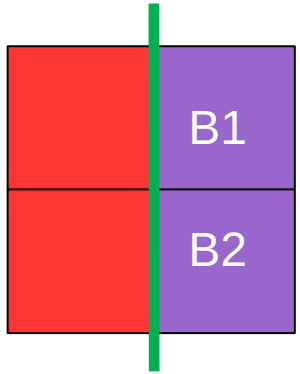
- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Parallel Coarsening

**Special case**:

The parent cell should be computed on Processor A

Some fine cells that A needs are on Processor B

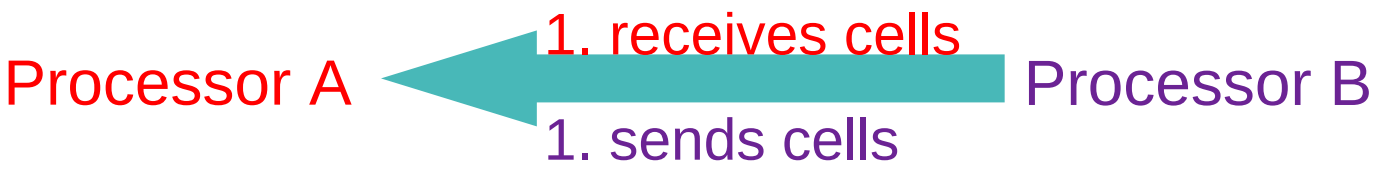



Processor B

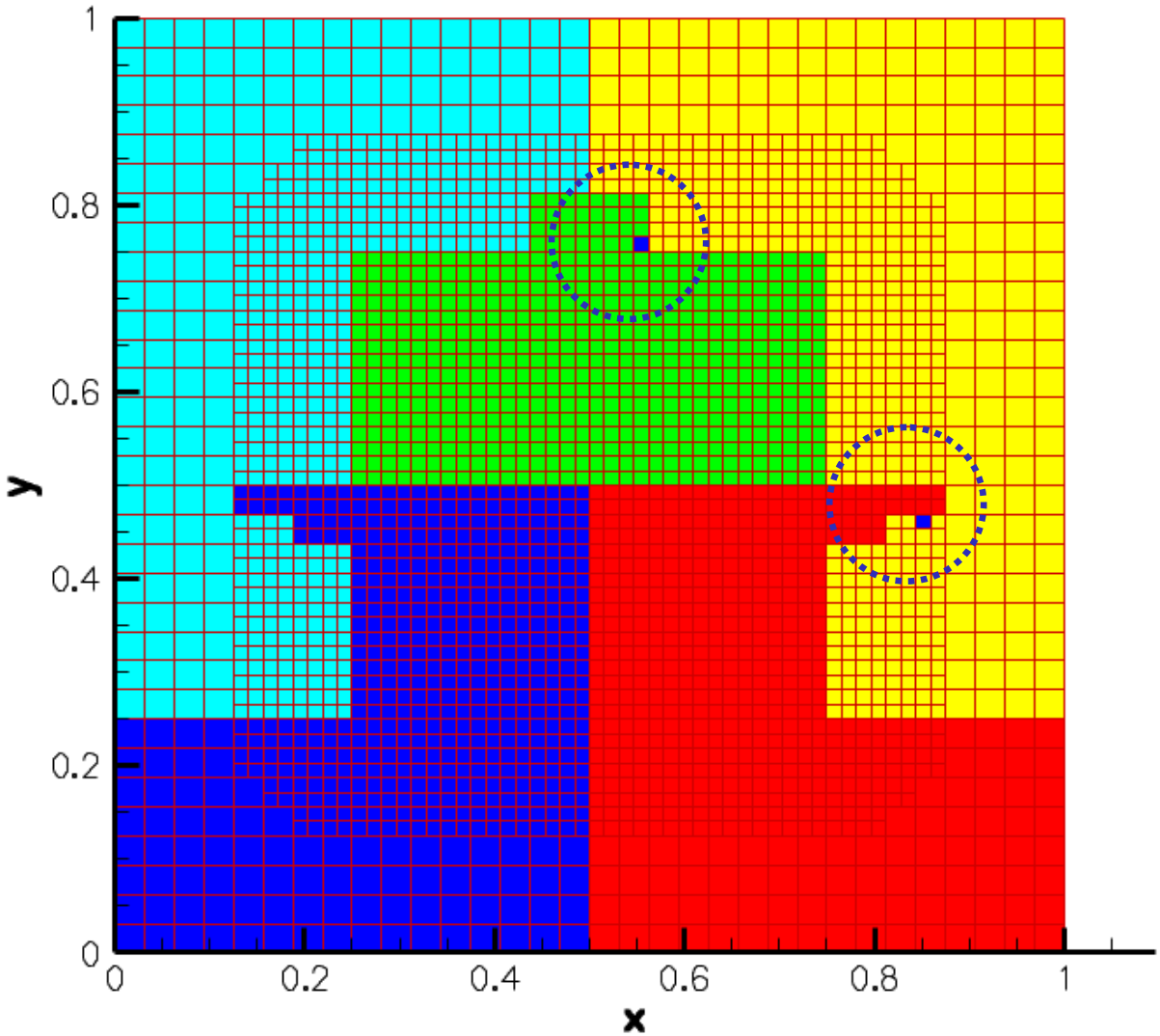
- runs through its data on level j+1
- determines the parent of the cells B1, B2
- the parent should be on Processor A
- he sends cells B1, B2 to Processor A, without waiting for a request

Processor A

- accepts the data from Processor B
- doesn't send any request



# Parallel Coarsening




Processor 0

Processor 1

Processor 2

Processor 3

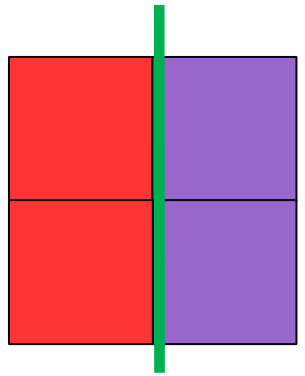
Processor 4

Cells to be transfered at processor boundary

# Details Computation

The detail of one cell should be computed on Processor A

Some fine cells that A needs are on Processor B



Partitions boundary

1.requests cells

1.accepts requests

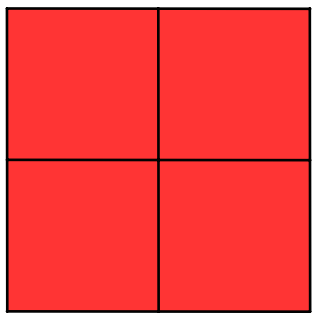
Processor A

Processor B

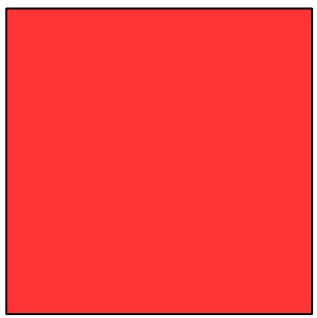
2. receives cells

2. sends requested cells

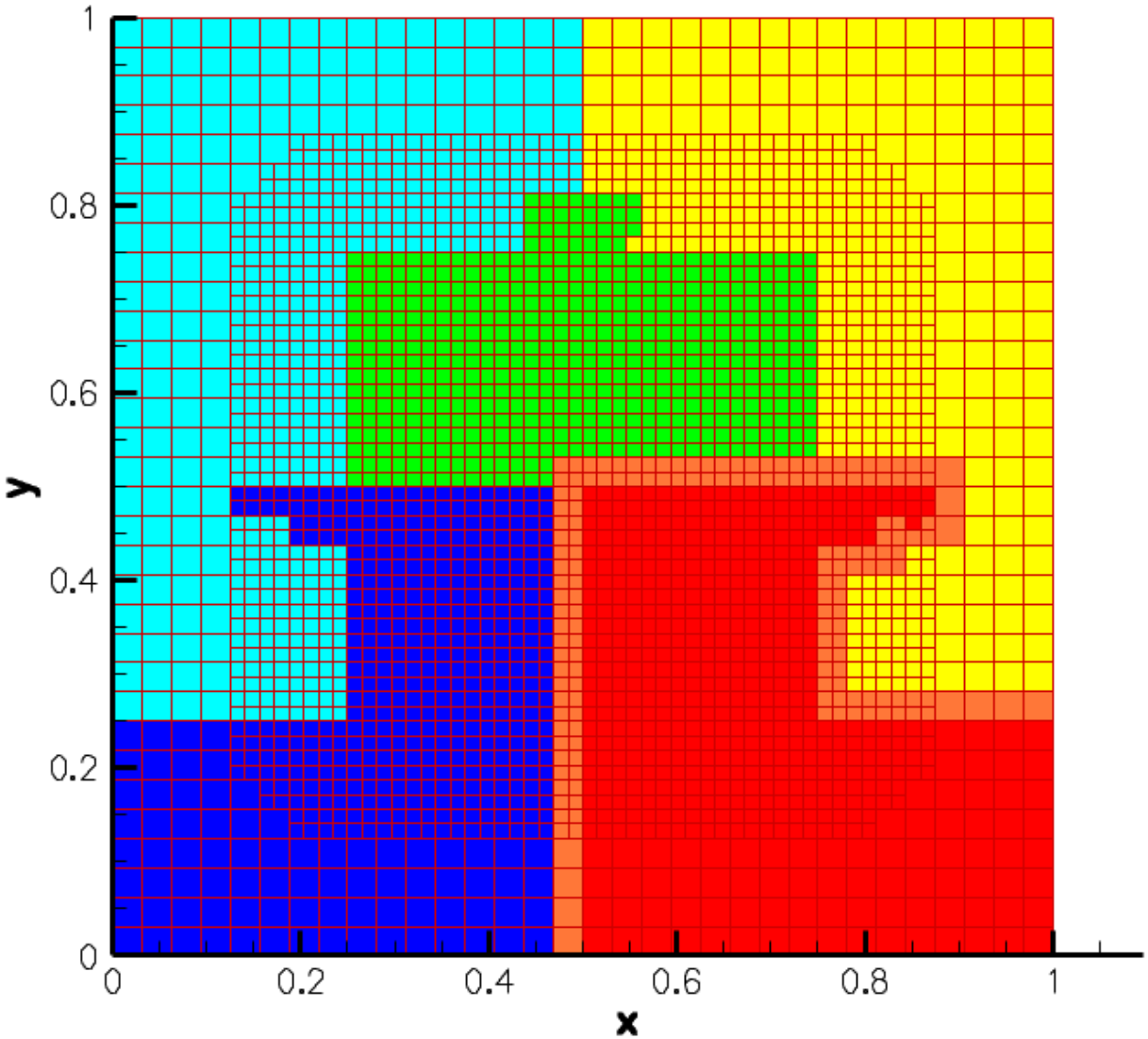
Processor A



Compute details



# Details Computation




Processor 0

Processor 1

Processor 2

Processor 3

Processor 4

Boundary data to send to processor 4

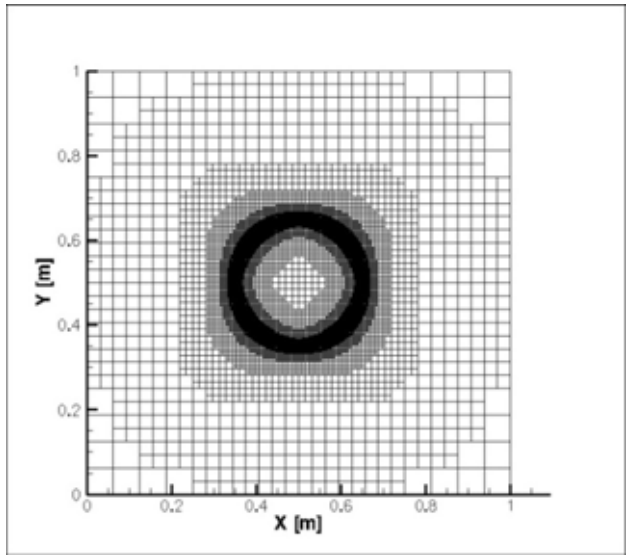
# MST Performance studies

**Input**: locally refined grid with:

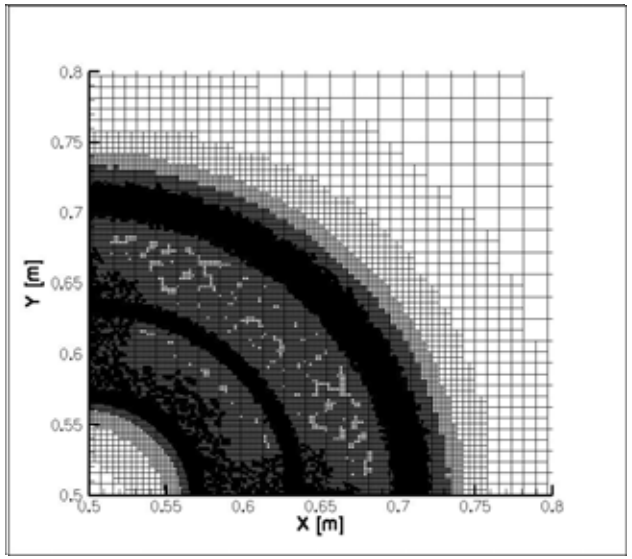
- number of refinemen levels: L = 10
- number of cells: 437236
- coarsest grid dimension: 8 x 8 (cells)

| No Procs | Mst time | Transfer time | Data sent | Data received | Initial workload |
|---|---|---|---|---|---|
| 1 | 10.2158 | 0 | 0 | 0 | **437236** |
| 2 | 5.55433 | 0.066706 | 2020 | 2020 | 218618 |
| 3 | 3.45852 | 0.170083 | 5985 | 5975 | 145746 |
| 4 | 2.81636 | 0.1258 | 2048 | 2048 | 109309 |
| 5 | 2.1364 | 0.127011 | 10826 | 10706 | 87448 |
| 6 | 1.78998 | 0.126228 | 7240 | 7190 | 72876 |
| 7 | 1.55737 | 0.133888 | 9097 | 9078 | 62464 |
| 8 | 1.47147 | 0.103048 | 6899 | 6954 | 54658 |
| 9 | 1.33229 | 0.105228 | 7112 | 7068 | 48588 |
| 10 | 1.23401 | 0.135434 | 8856 | 8791 | 43729 |
| 11 | 1.61369 | 0.511713 | 7744 | 7801 | 39756 |
| 12 | 1.02436 | 0.115268 | 5081 | 5113 | 36440 |

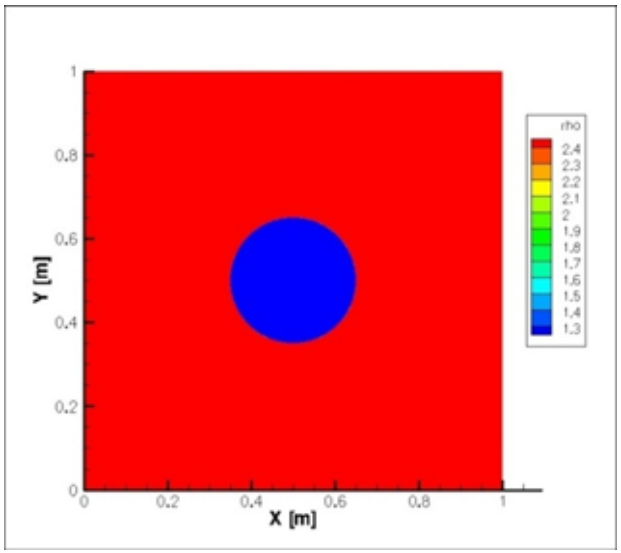
# 2D Implosion Problem



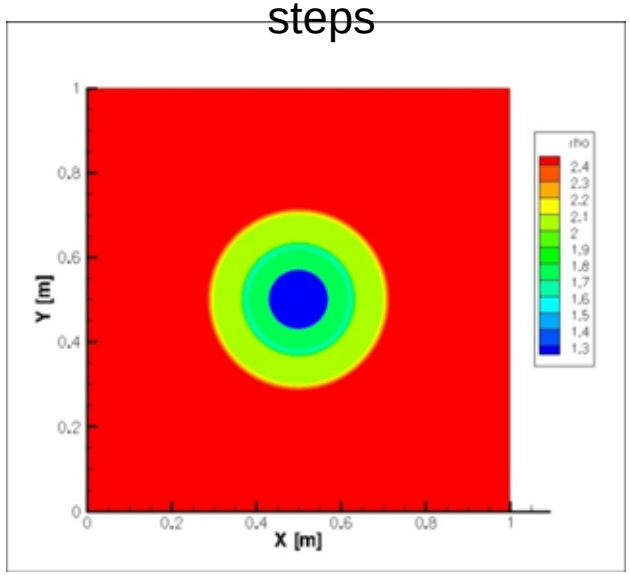

Initial adaptive grid




Adaptive grid after 3200 time steps




Initial density




Density after 3200 time steps

# Performance studies

**Input**: locally refined grid with:

- number of refinement levels:
  L = 10
- no. of cells on starting grid:
  193972
- no. of cells after 100 iterations:
  563980
- final number of cells:
  760264 (after 3200 iterations)
- coarsest grid dimension:
  8 x 8 (cells)

| No Procs | Time/3200 iterations (minutes) | Time/100 iterations (minutes) | Initial workload / processor |
|---|---|---|---|
| 1 | 631.55 | 8.46 | 193972 |
| 2 | 494.58 | 5.39 | 96986 |
| 3 | 381.17 | 4.21 | 64591 |
| 4 | 275.24 | 3.10 | 48493 |
| 5 | 260.16 | 2.36 | 38795 |
| 6 | 241.50 | 2.34 | 32329 |
| 7 | 219.33 | 2.24 | 27711 |
| 8 | 220.17 | 1.52 | 24247 |
| 9 | 210.57 | 1.42 | 21553 |
| 10 | 190.30 | 1.34 | 19398 |
| 11 | 176.48 | 1.31 | 17634 |

# Outline

- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Summary

- The construction of the SFC at runtime
- Data partitioning and mapping to processors
- Parallel grid adaptation (2D, 3D)
- 2D implosion problem

# Outlook

- Strategy for multiblock parallel load balancing

- Application to the 3D wake vortex simmulation

# Outline

- Multiscale Grid Adaptation
  - General setting
  - Multiscale analysis: encoding and decoding
- Data Partitioning
  - Locally refined grids
  - Space Filling Curve construction
  - Load balancing and data distribution to processors
- Parallel Encoding
  - Coarse cells averages computation
  - Details computation
  - Performance studies
- Summary & Outlook

# Acknowledgements

Prof. Dr. Wolfgang Dahmen

Prof. Dr. Siegfried Müller


Thank you for your attention!