Parallel streamline tracking for UnTRIM

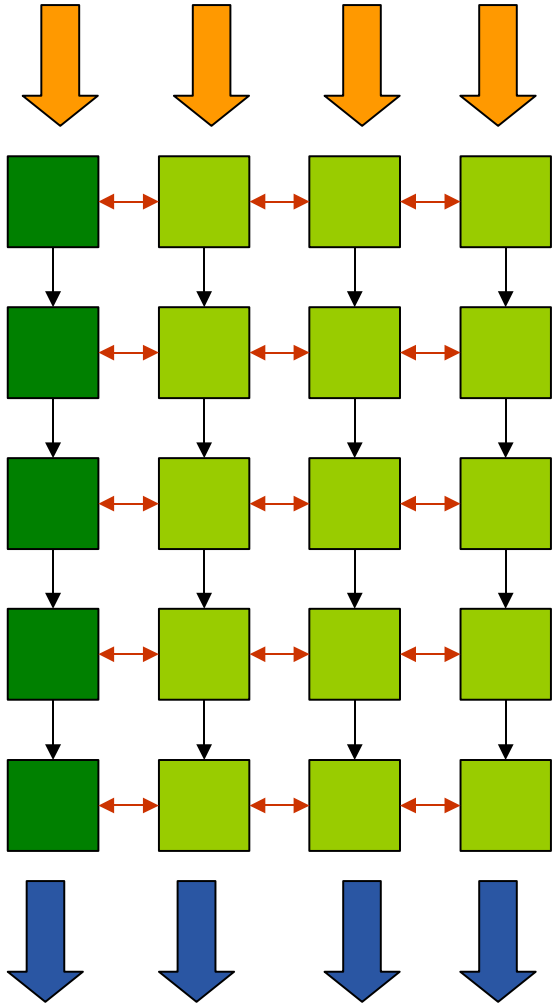
Jacek A. Jankowski

BAW
Department Inland
Waterways Engineering

Fourth UnTRIM Workshop
Trento, 7-9 May 2007

Message-passing parallelism

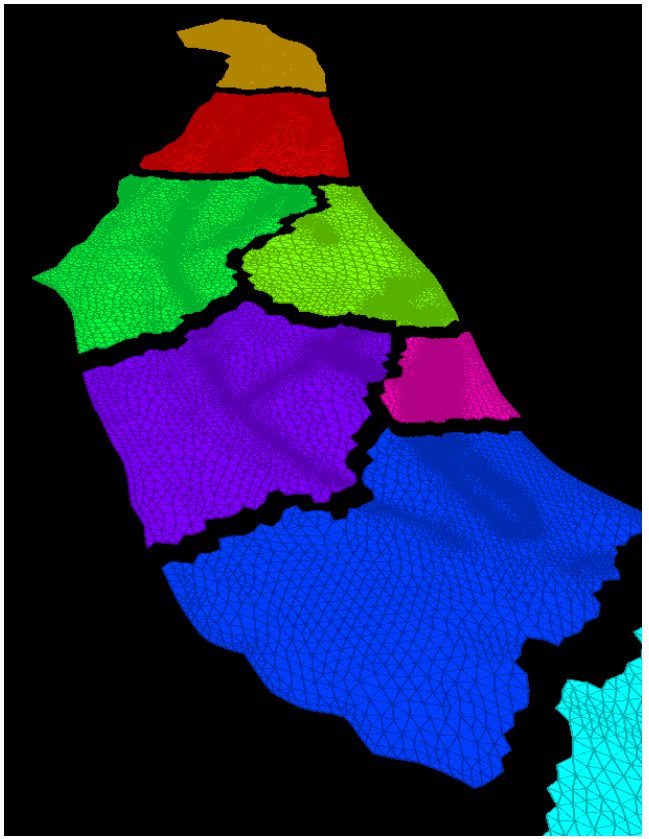
- Each processor executes a program copy with its own data
- Communication limits the scalability of the code
  - preparing data for sending
  - communication *itself*
  - integrating the received data



```
mpirun -np 4 ./prog.exe
```

Domain decomposition method

- Parallel implementation with domain decomposition and overlapping mesh partitions
- This leads to point-to-point communication between neighbouring partitions
- Semi-Lagrangian advection methods do not fit well to this scheme



Contents

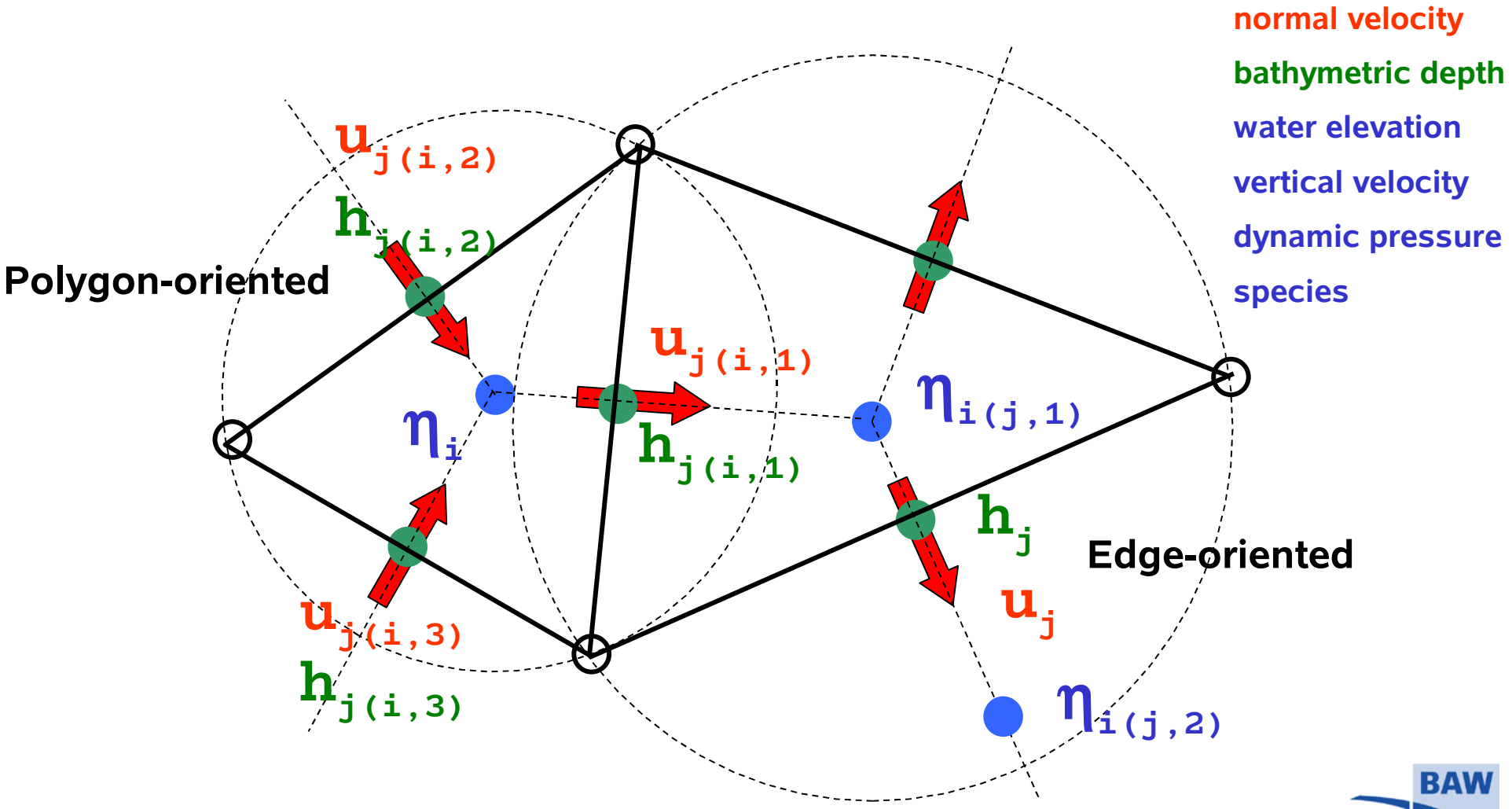
- Communication
  - point-to-point for the halo swapping (done 2004-5)
  - global communication for the advection (done 2007)

- *Recommendations – how to*
  - *reduce differences between serial and parallel results*
  - *maintain scalability*

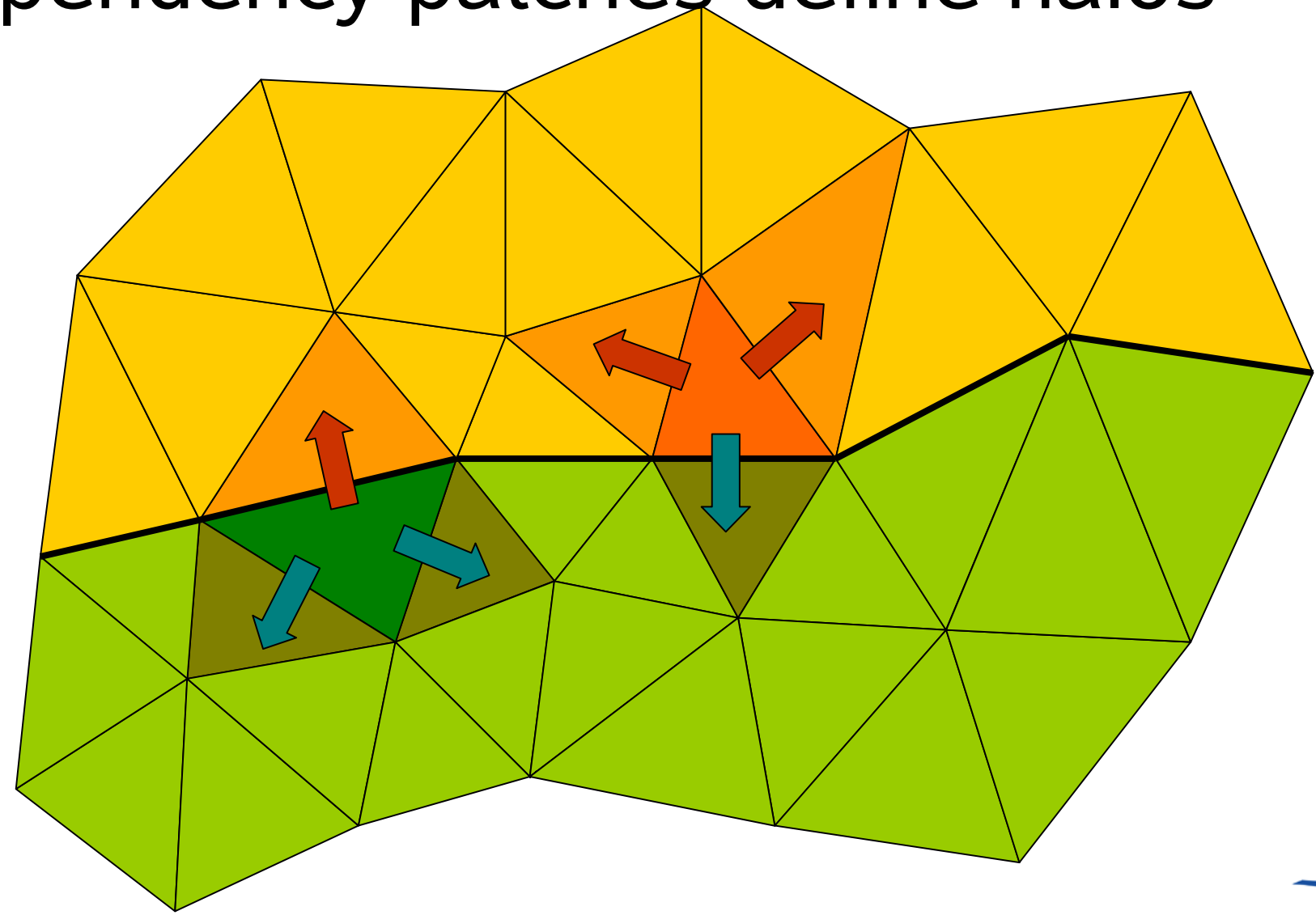
Point-to-point communication

Dealing with the Finite Volume/Differences Methods

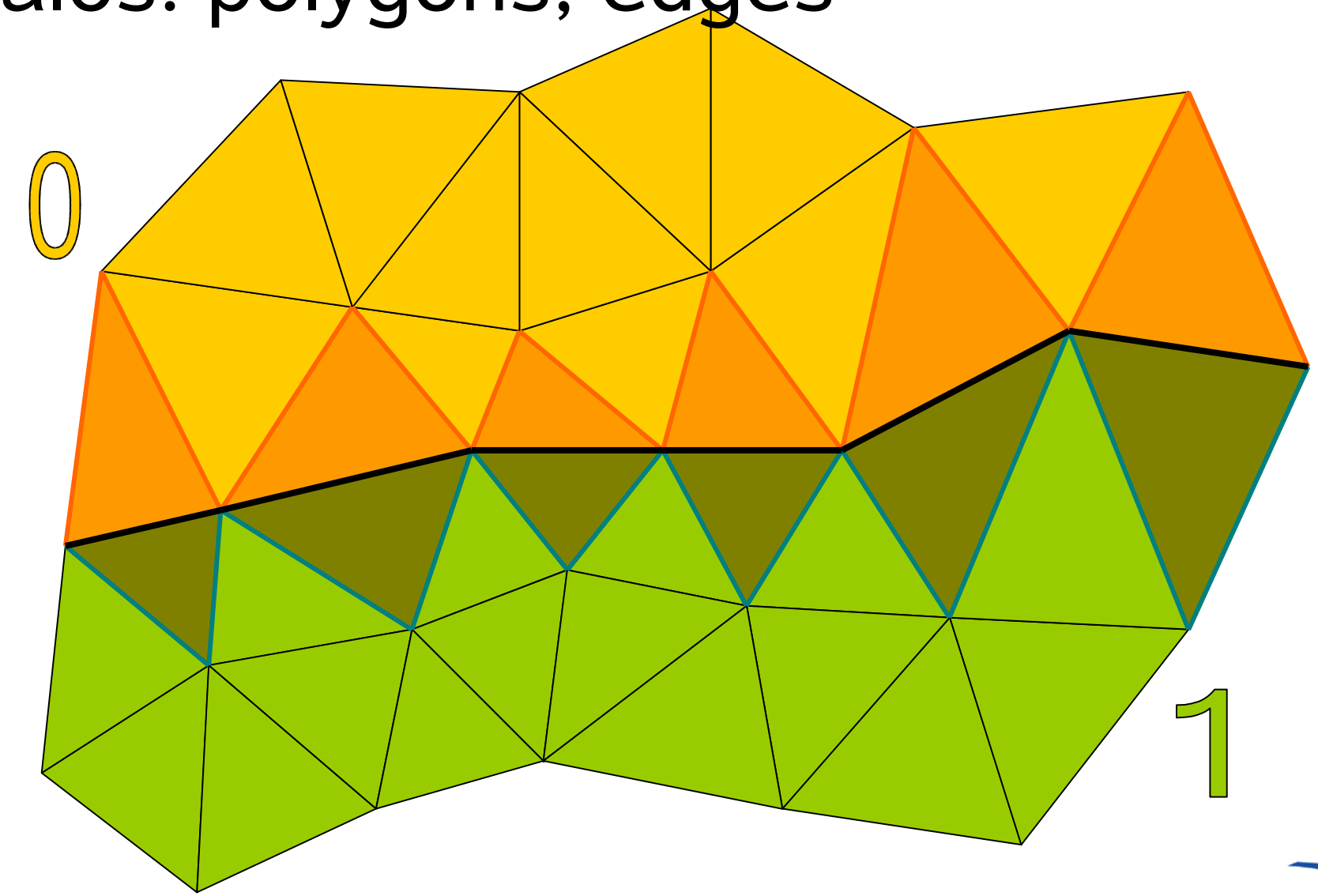
Horizontal structure



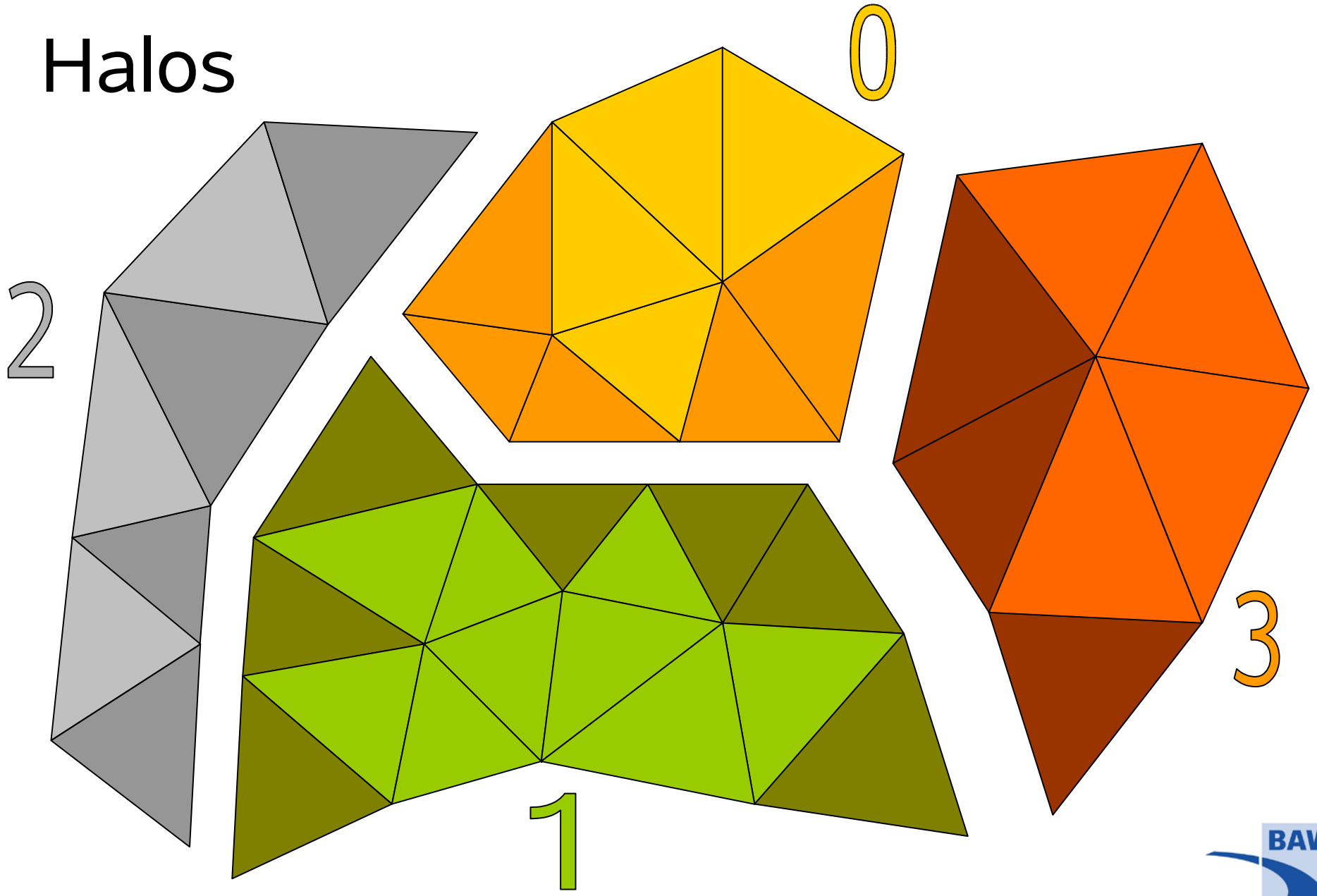
Dependency patches define halos



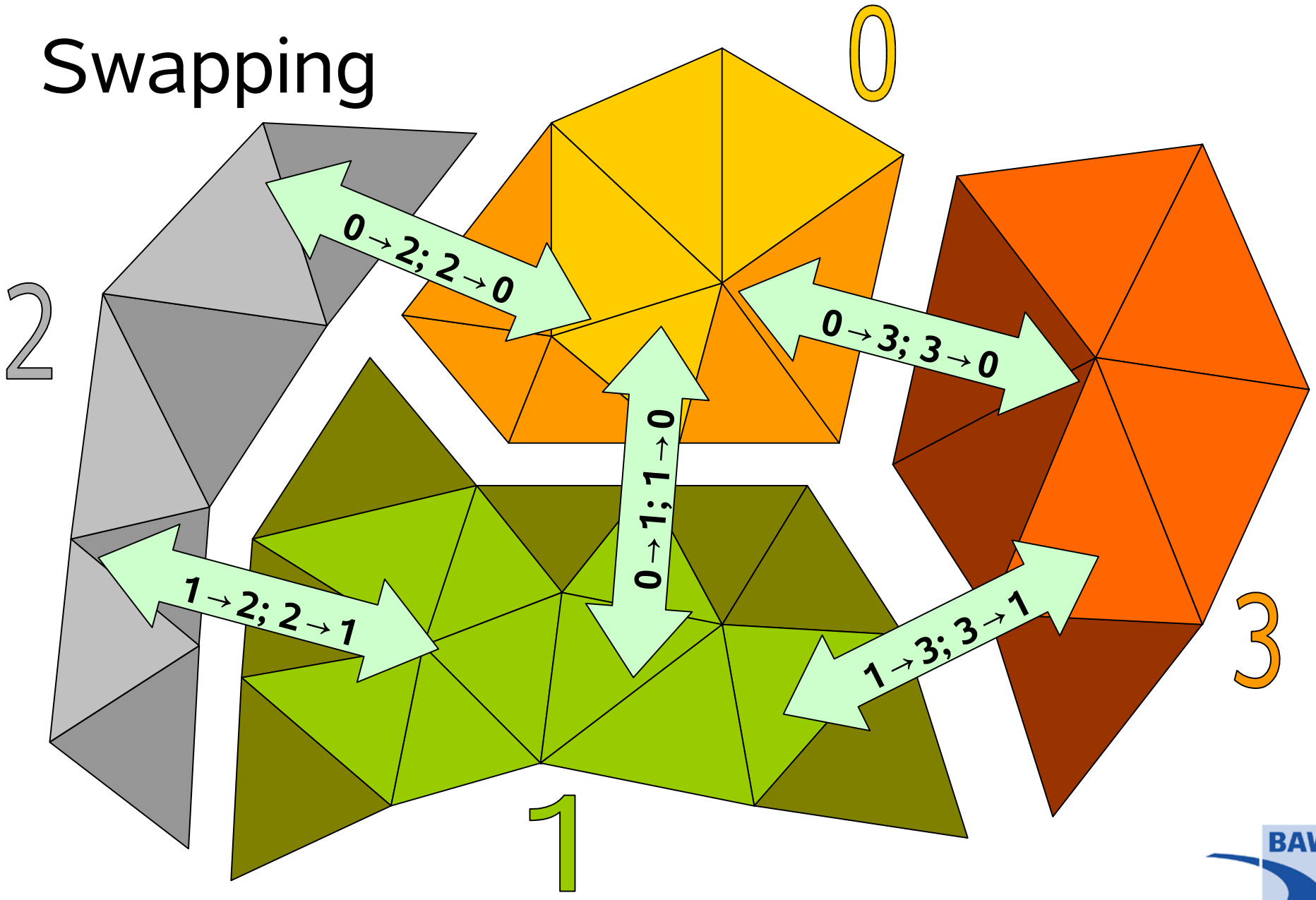
Halos: polygons, edges



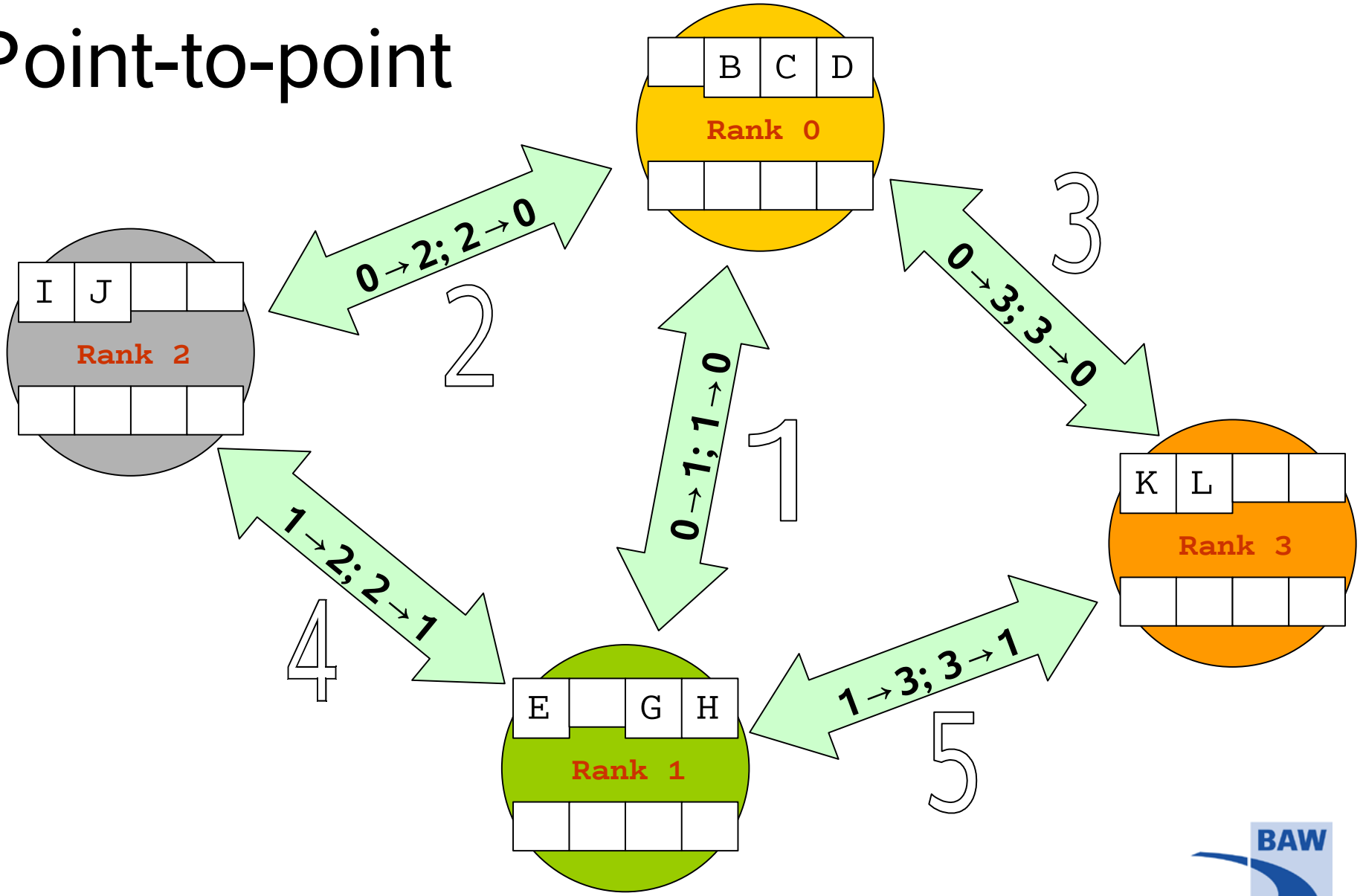
Halos



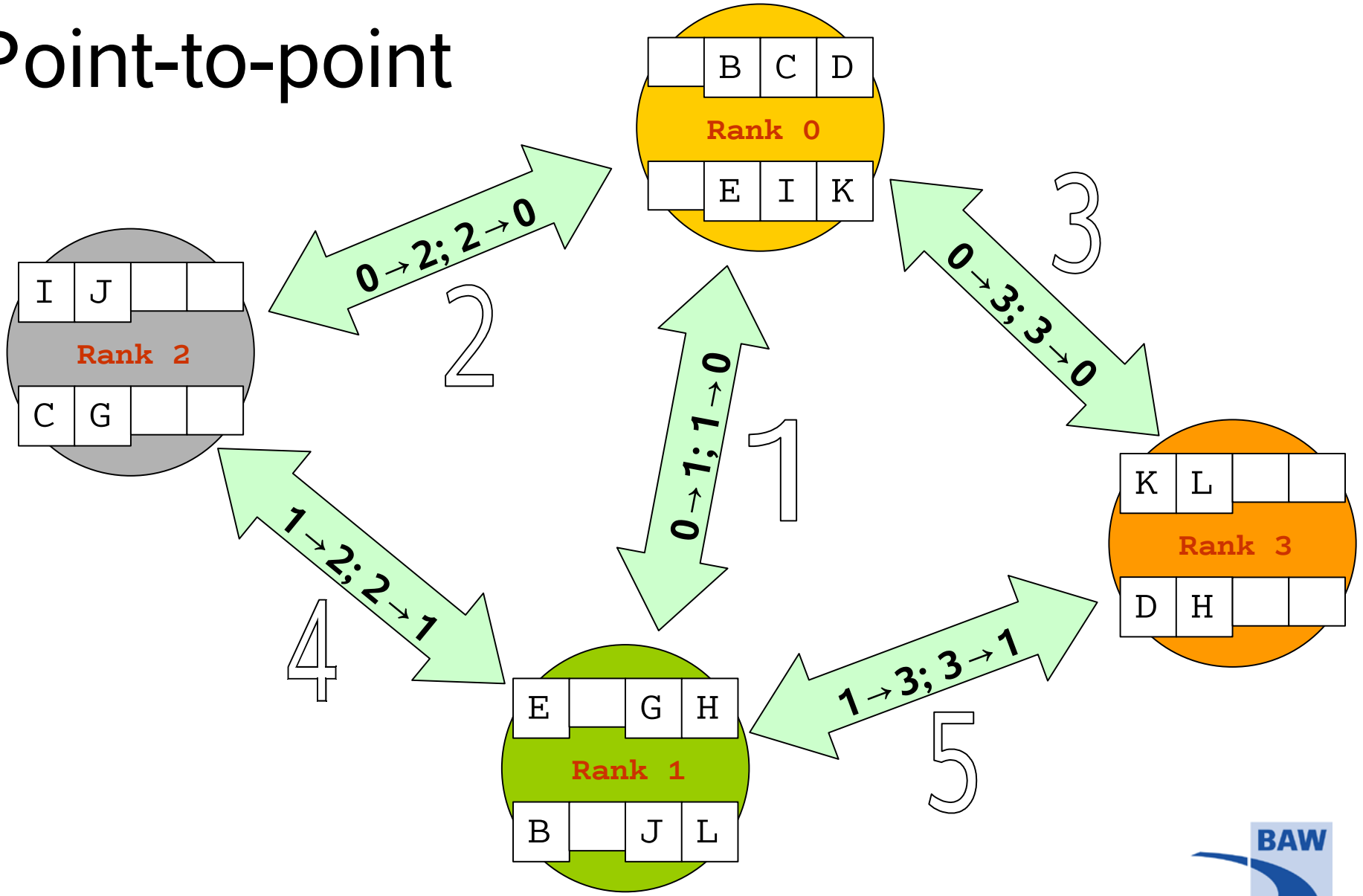
Swapping



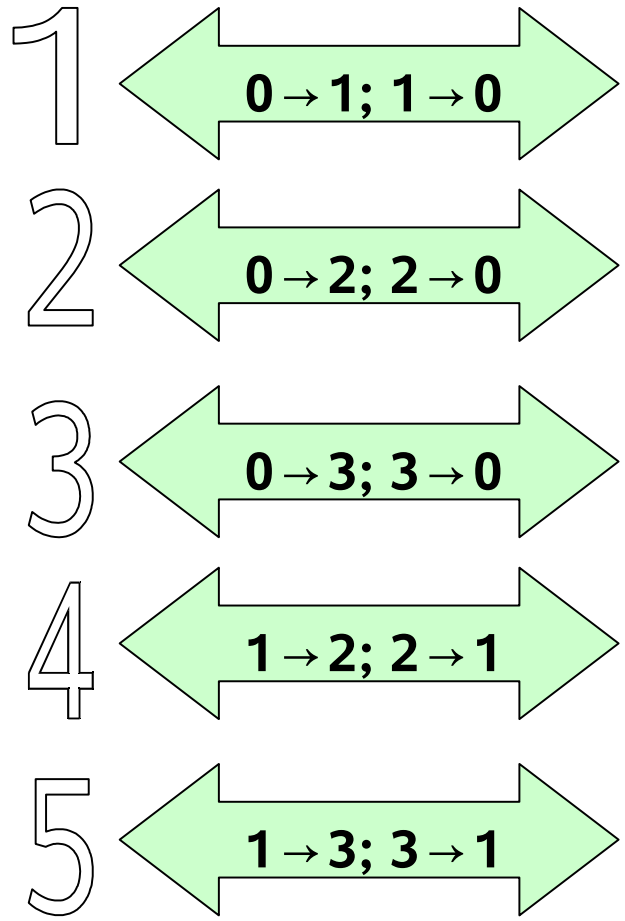
Point-to-point



Point-to-point

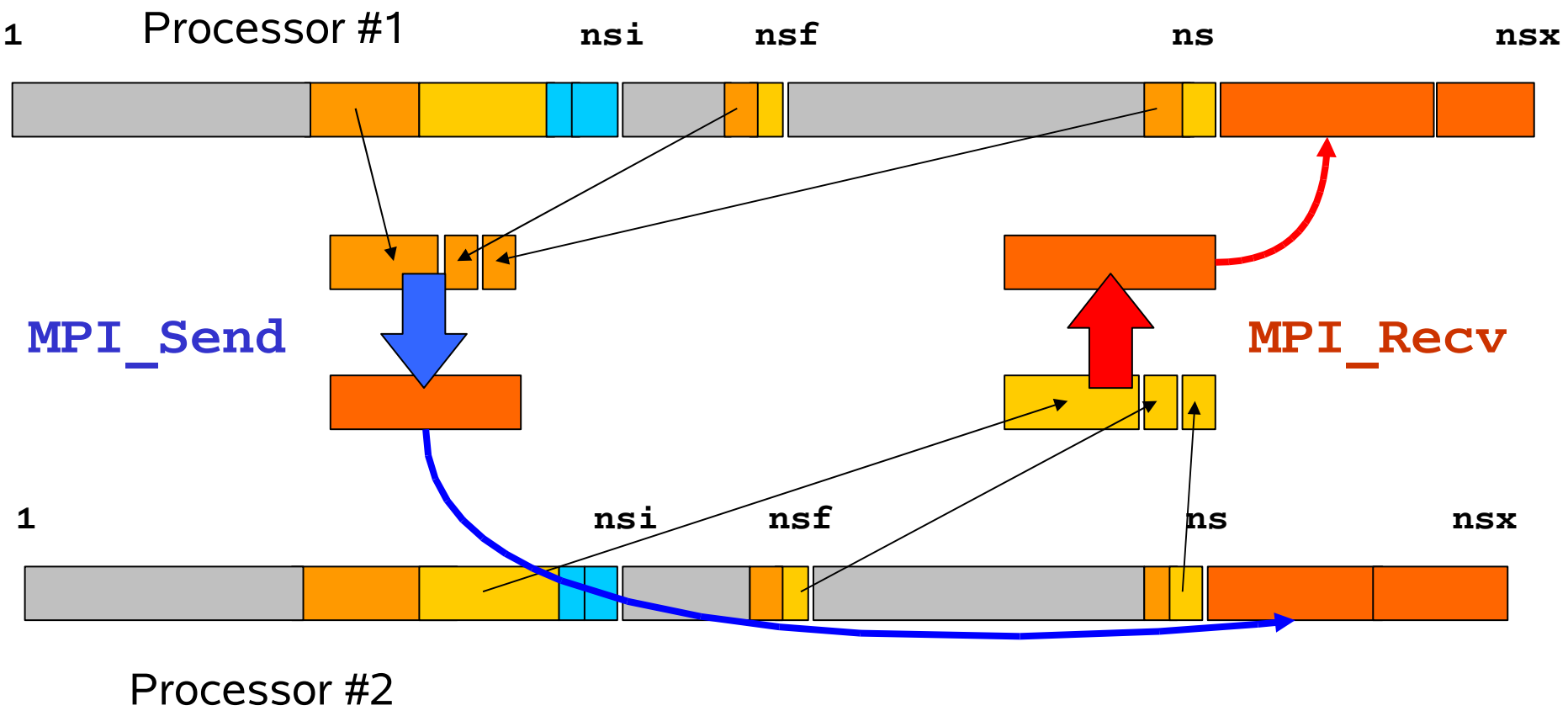


MPI_SendRecv

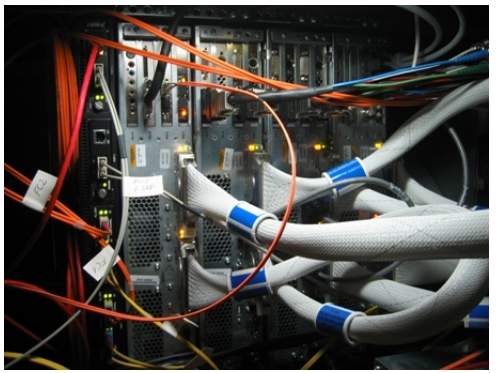


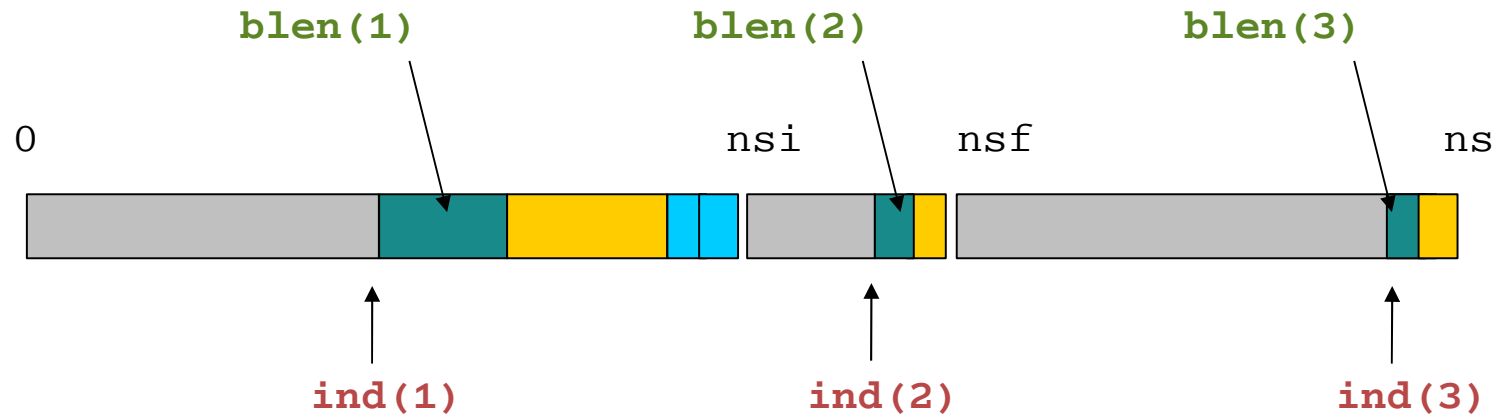
- Point-to-point communication
- Halo swapping in the order of direct neighbour pairs
- Objects are polygons, edges, cells, faces…

`MPI_SendRecv` with Buffers



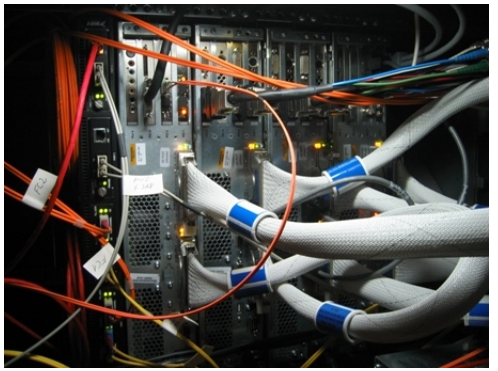
MPI_Type_Indexed






```
CALL MPI_Type_Indexed (3,blen,ind,MPI_INTEGER,newtype,ier)
CALL MPI_Commit (newtype, ier)
CALL MPI_Send (field,1,newtype,...)
CALL MPI_Free (newtype,ier)
```

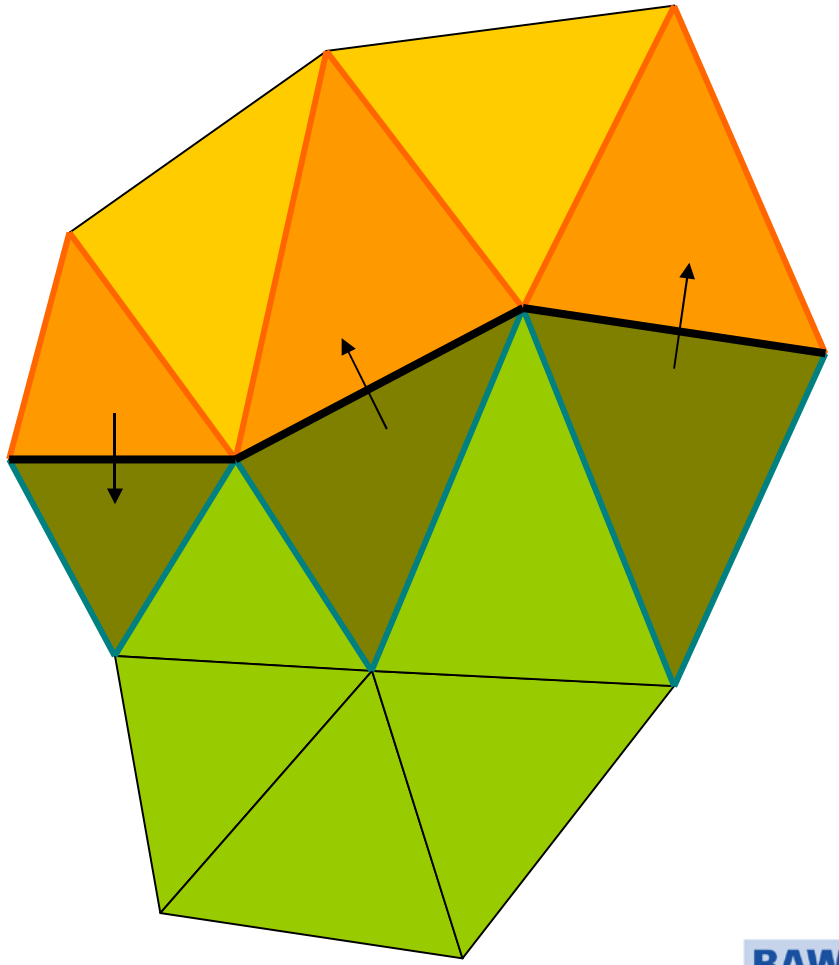
FD/FV Method: Swapping



- FD/FV numerical scheme of UnTRIM - point-to-point communication between neighbour partitions is adequate:
  - allocation of separate communication buffers
  - **send**: data sorted to a *send*-buffer and sent
  - **receive**: data copied from a *receive*-buffer
- Performance improvements(?): MPI Data types, pointers

Common edges / faces treatment

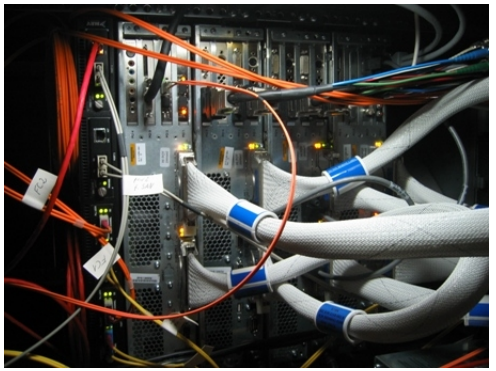
- Values computed on common objects available for both partners
- Options – mean value, leaving “as is”, upstream…
- Best results: taking the upstream value



Global communication

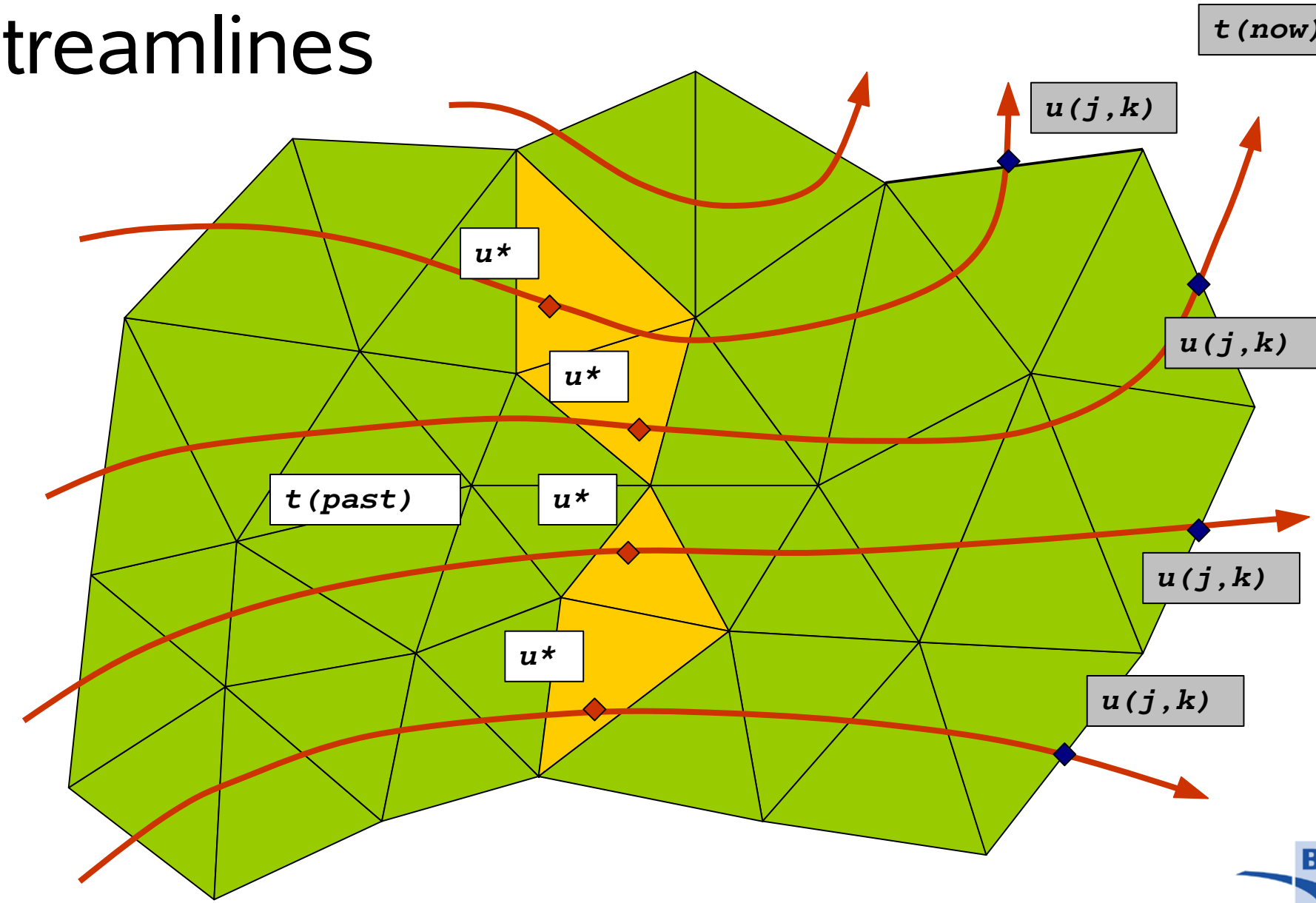
Dealing with the Lagrangian methods (advection)

Lagrangian scheme

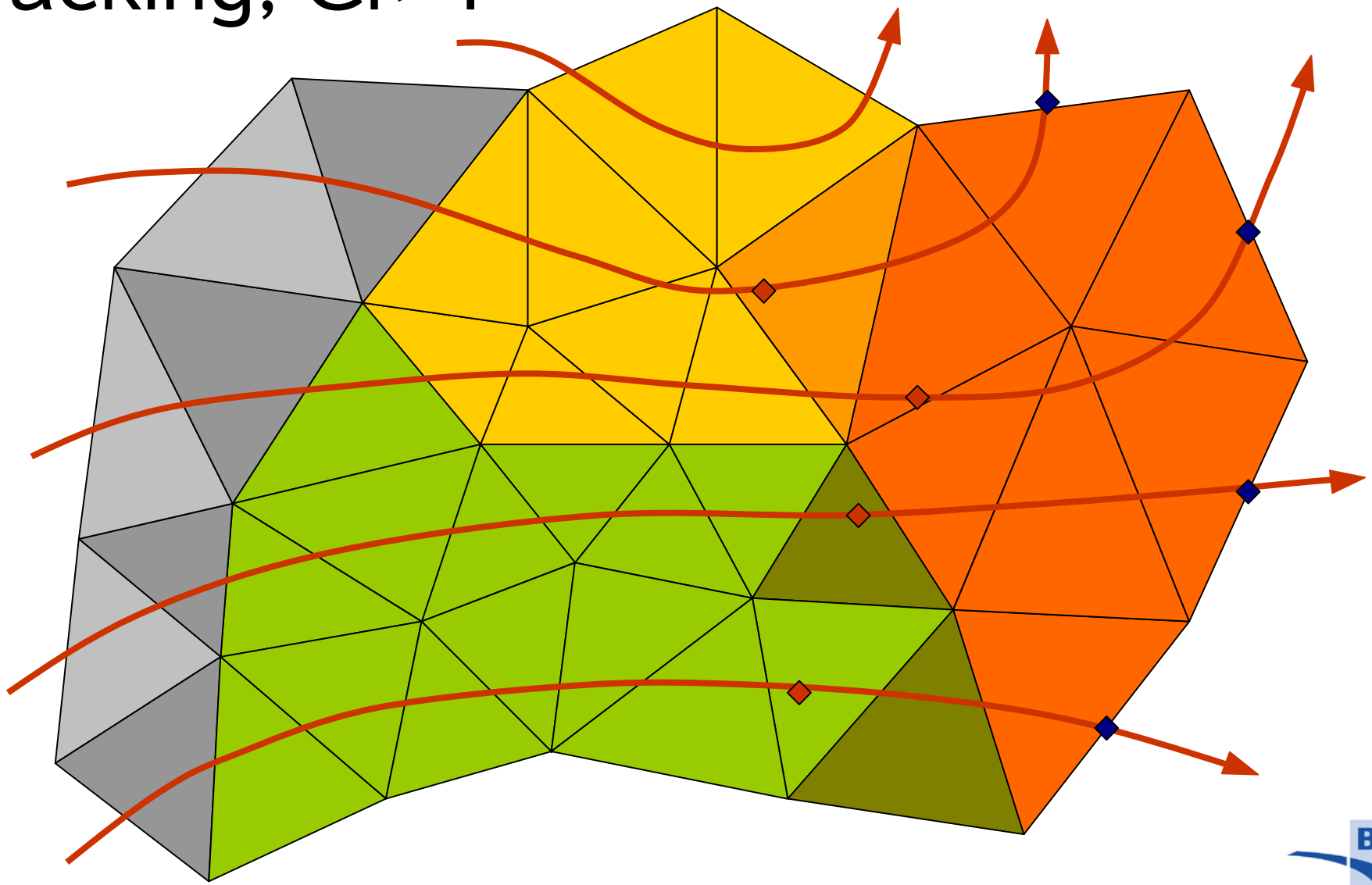


- Advection scheme in UnTRIM – semi-Lagrangian
  - streamline tracking over the mesh – backward in time
  - interpolating a value at a found point in the mesh
  - applying the find value further on
- **semi²** - Requires a special treatment in partitioned meshes

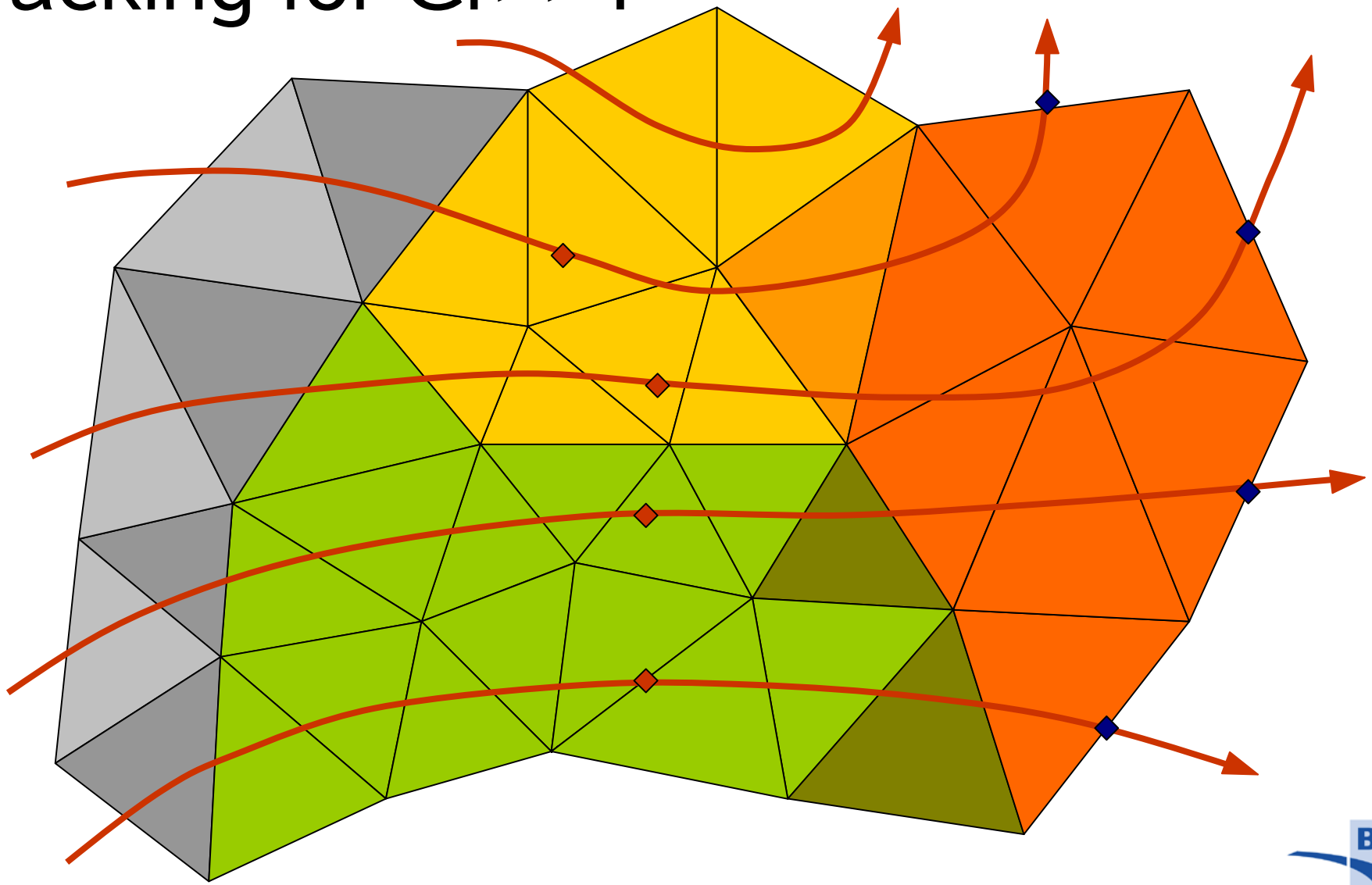
Streamlines



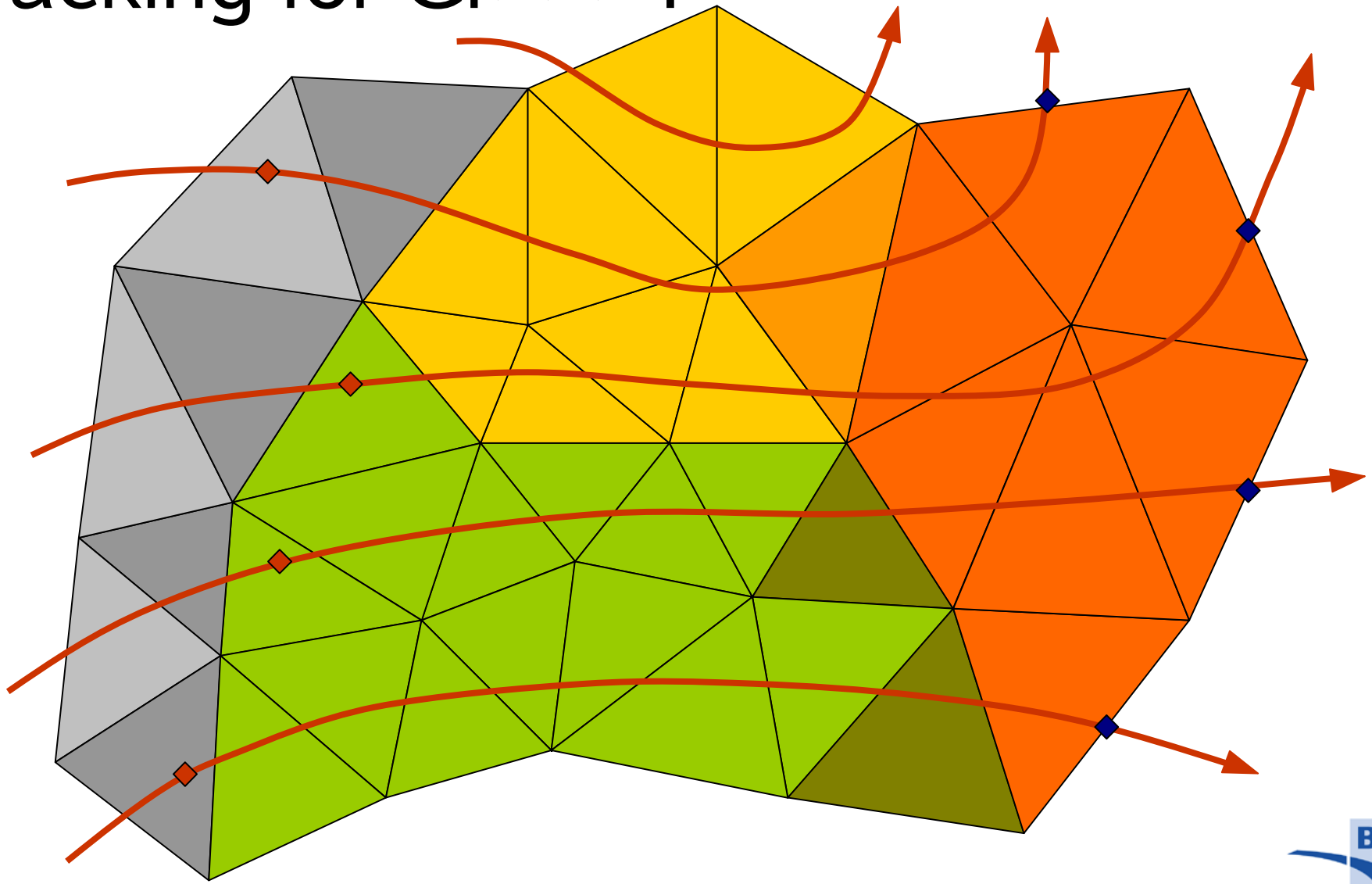
Tracking, Cr>1



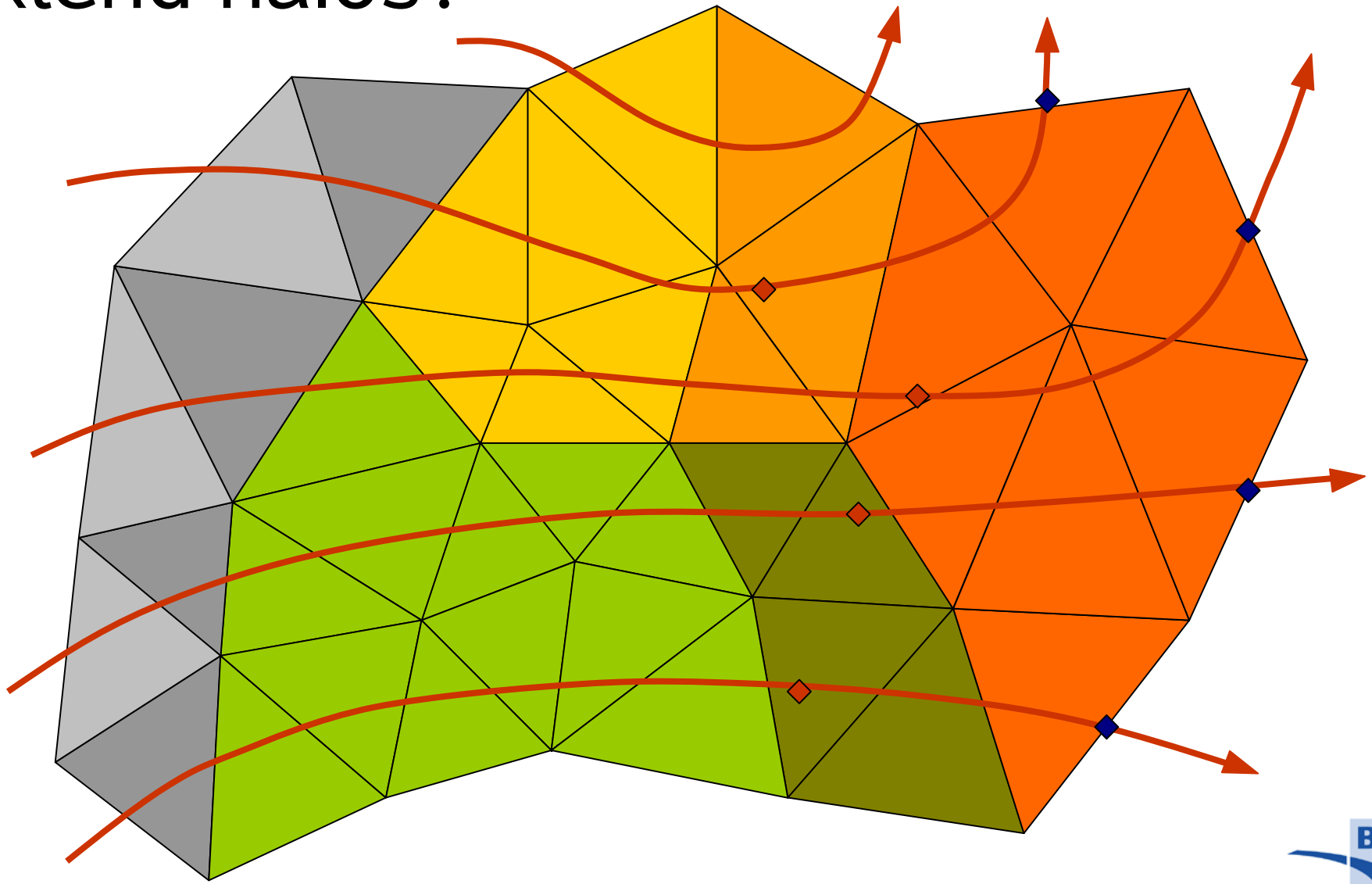
Tracking for Cr>>1



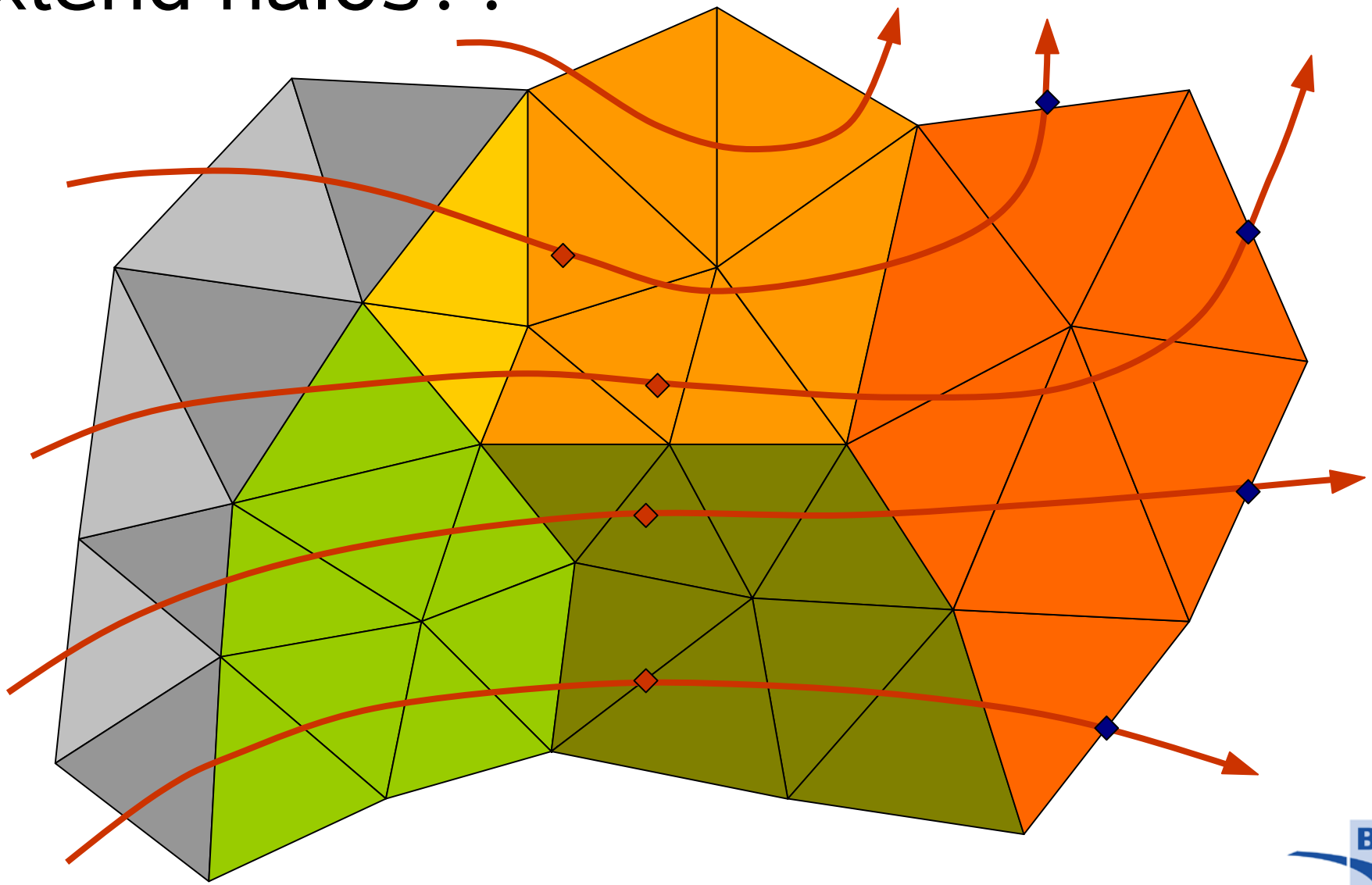
Tracking for Cr>>>1



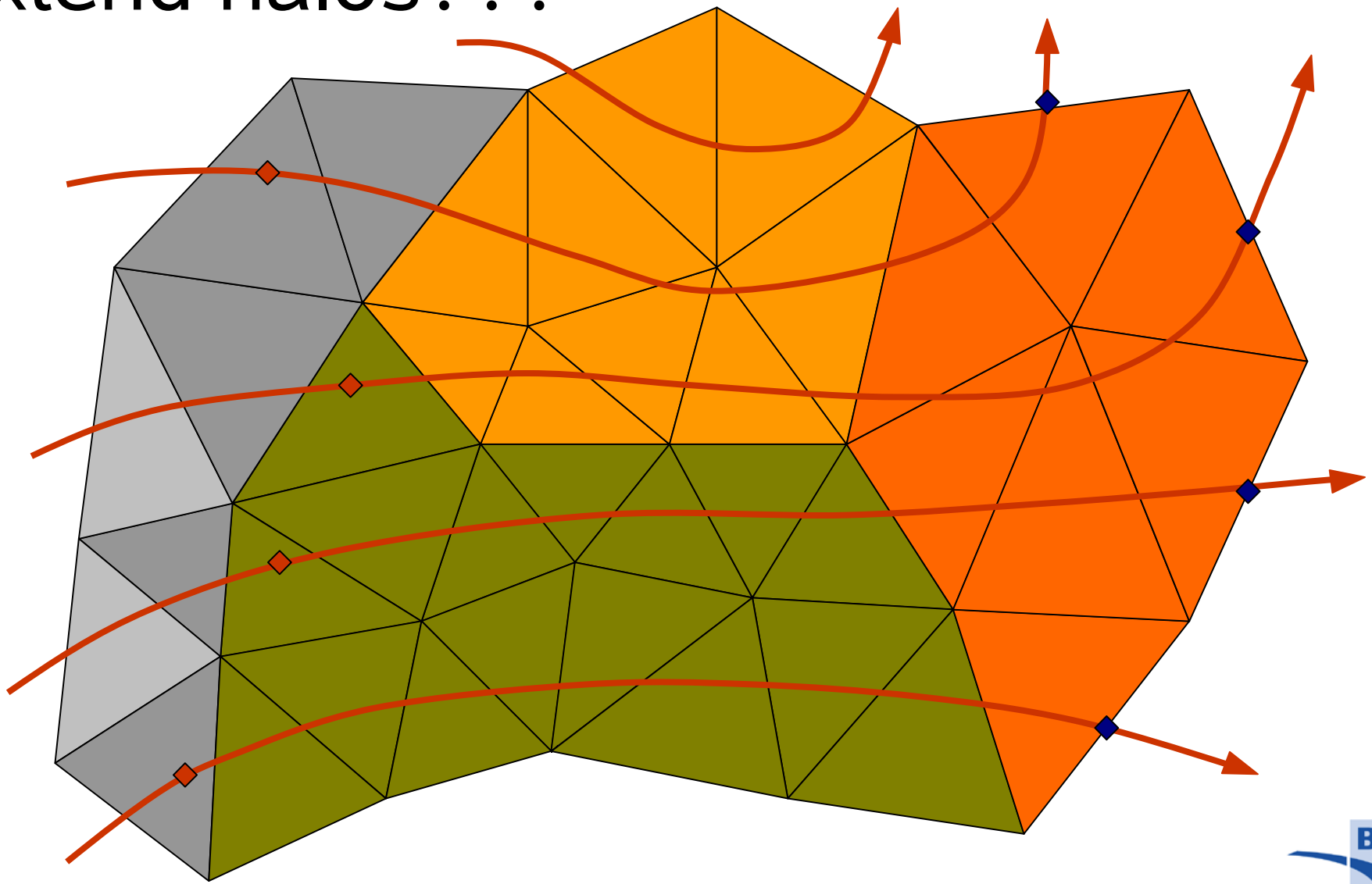
Extend halos?



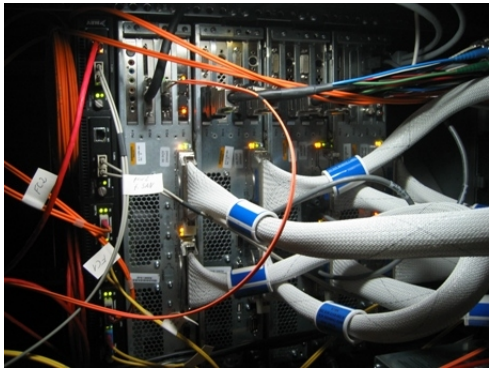
Extend halos??



Extend halos???



Tracking over partitions



- Streamline tracking is awkward in the point-to-point communication pattern between direct neighbours
- Inefficient for larger Courant numbers (large halos, further neighbours to communicate with…)
- Solution: Tracebacks leaving partitions treated as **separate objects** in an *autonomous* algorithm

Source: dogtagshop.de

Dog tags for lost tracebacks

An object describing a 'lost' traceback:

```
TYPE charac_type
   INTEGER  :: mypid,ior,jor,kor
   INTEGER  :: nepid,i,k
   REAL(dp) :: tres,xs(2),zs
   REAL(dp) :: us(2),ws
   INTEGER  :: isat,mem
END TYPE
```

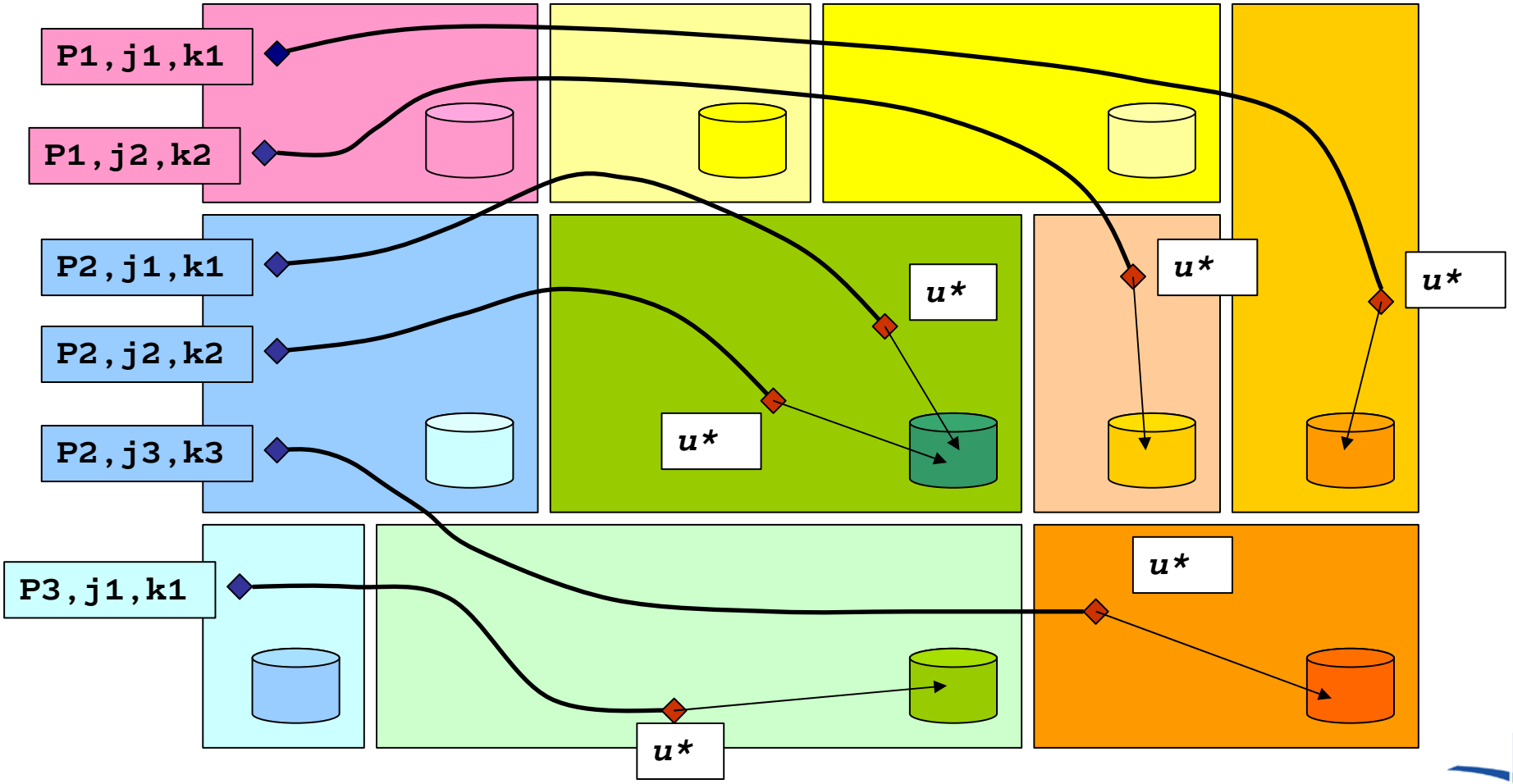
Autonomous tracking

- Define a **MPI data type** for a traceback
- Each traceback leaving a partition gets
  - an **identifier** with the position of the *head*
  - and a **'basket'** to bring values back
- The traceback is **implanted** in the neighbourhood to be followed further for the remaining fraction of time **`tres`**

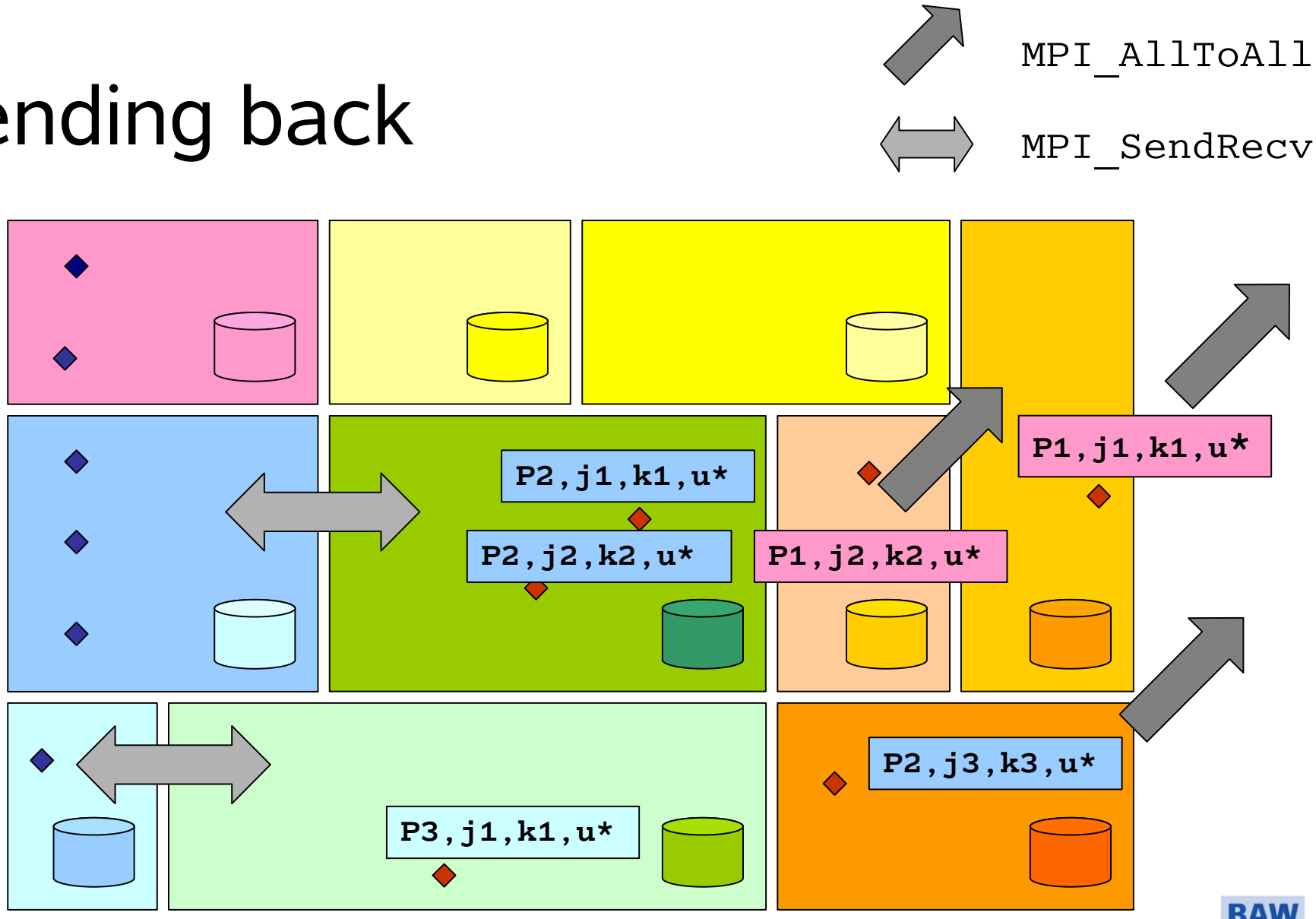
Autonomous tracking

- When the *foot* is found, the traceback is collected locally together with the interpolated value `u*` on a **heap**
- When all *feet* are found everywhere, the tracebacks are **sorted** according to the partition they originate from (*head*)
- ...and send back using the global communication methods
- ...and the found values applied

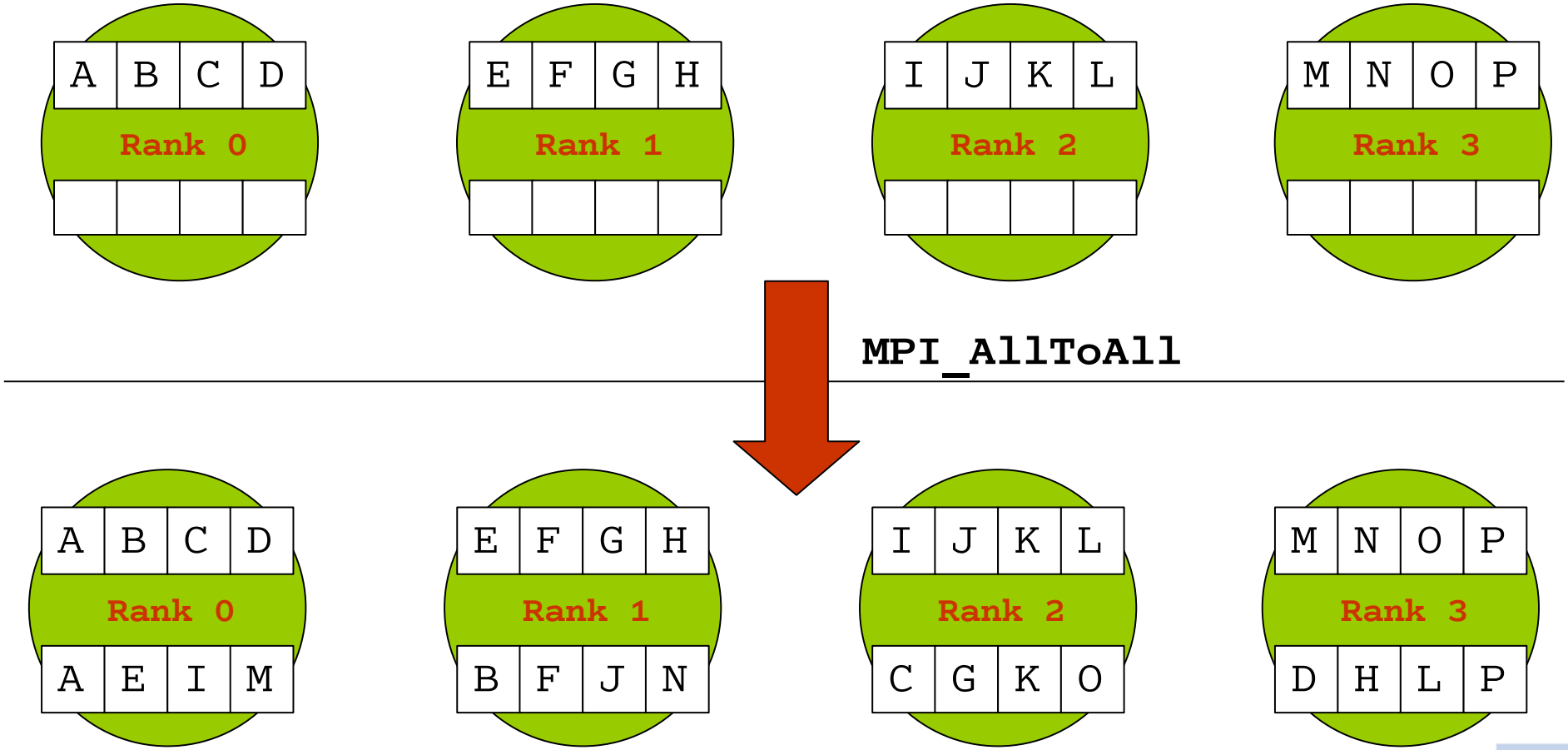
Autonomous tracking



Sending back



MPI_AllToAll

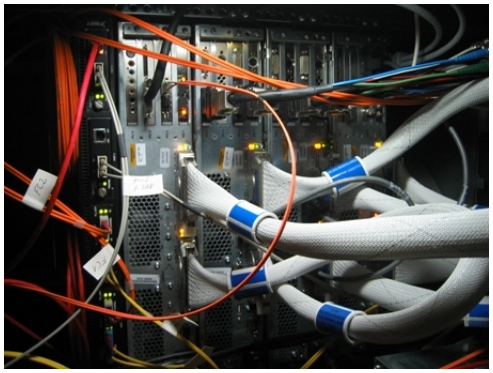


`MPI_AllToAll` contra `MPI_SendRecv`

- Initially, *point-to-point* communication has been chosen whenever possible, **`MPI_Send_Recv`** in order to reduce the number of communication partners

- Test have shown that the *global* communication is efficient enough - finally **`MPI_AllToAll`** and **`MPI_AllToAllv`** has been applied for all sorts of communication

Summary: Communication



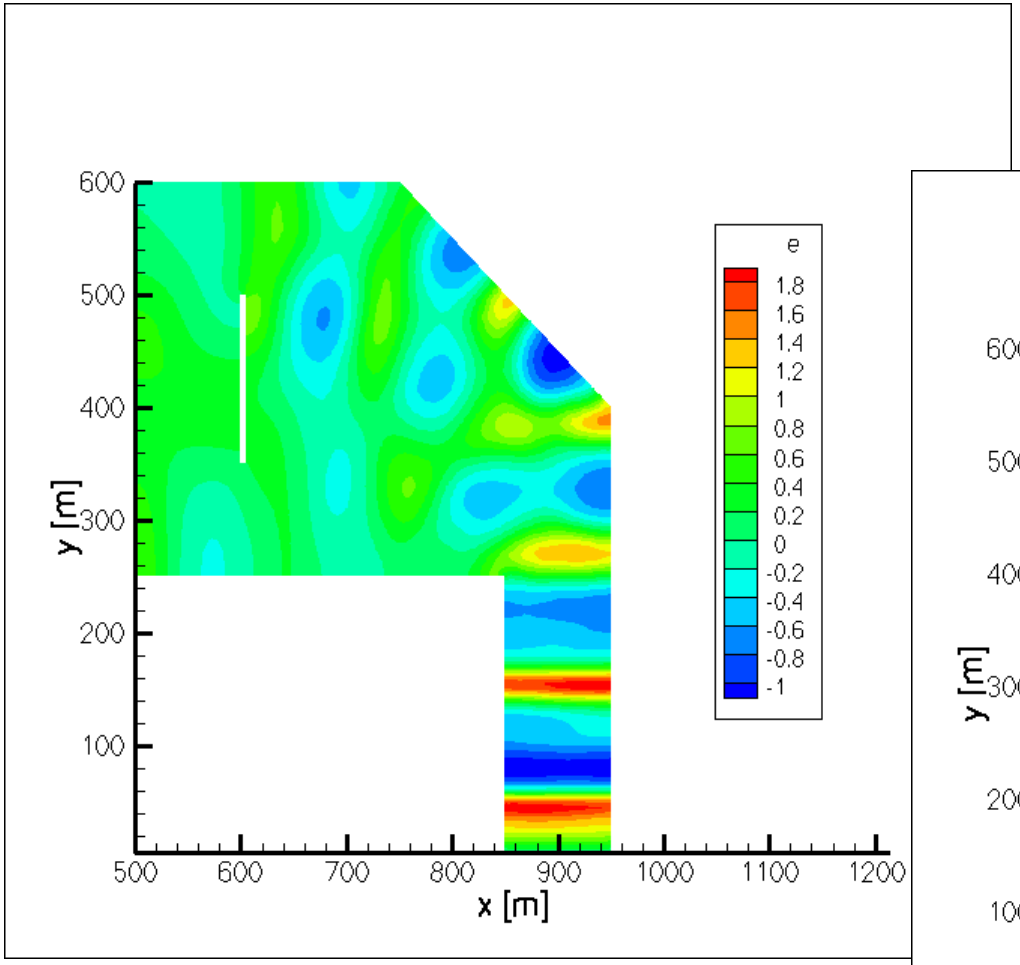
- FD/FV (Eulerian):
  - swapping halo values: point-to-point communication
  - common interface edges or faces: upstream values

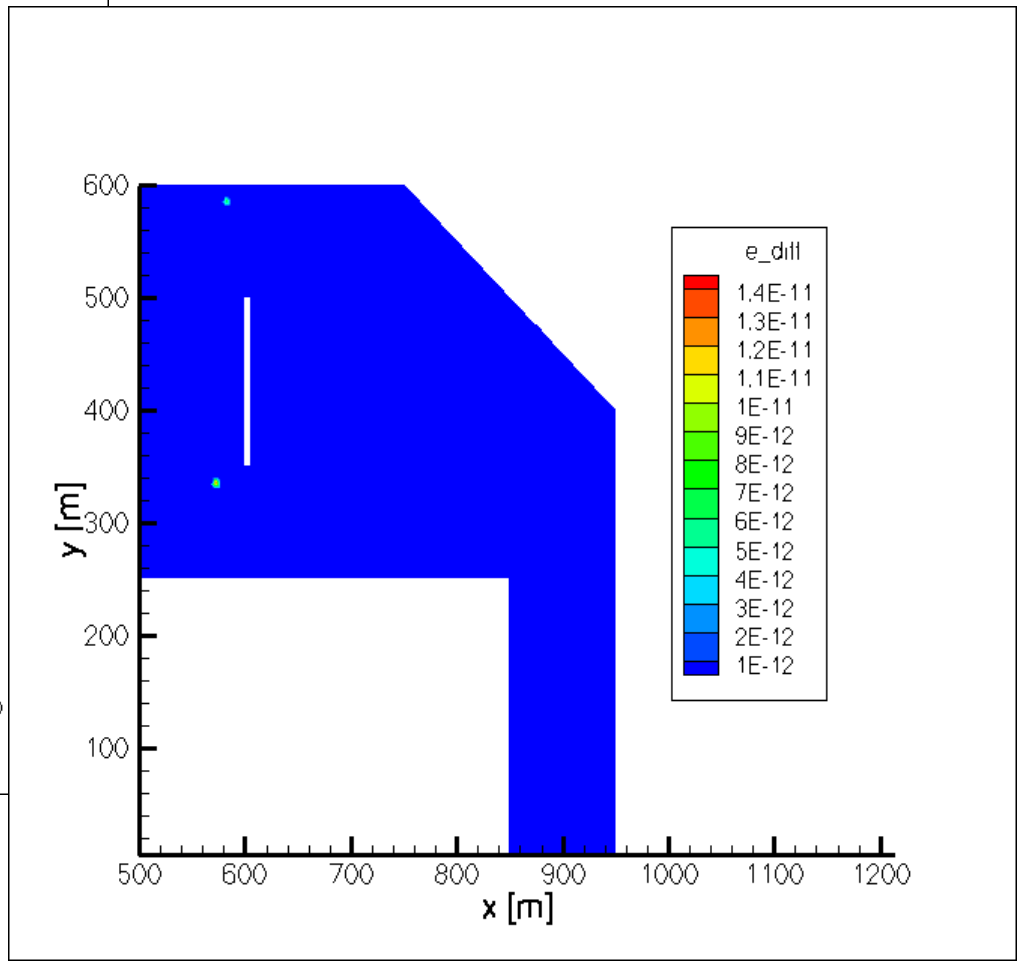
- Advection (Lagrangian):
  - streamline tracking treating tracebacks as autonomous objects: global communication

Verification

Differences in parallel and serial results

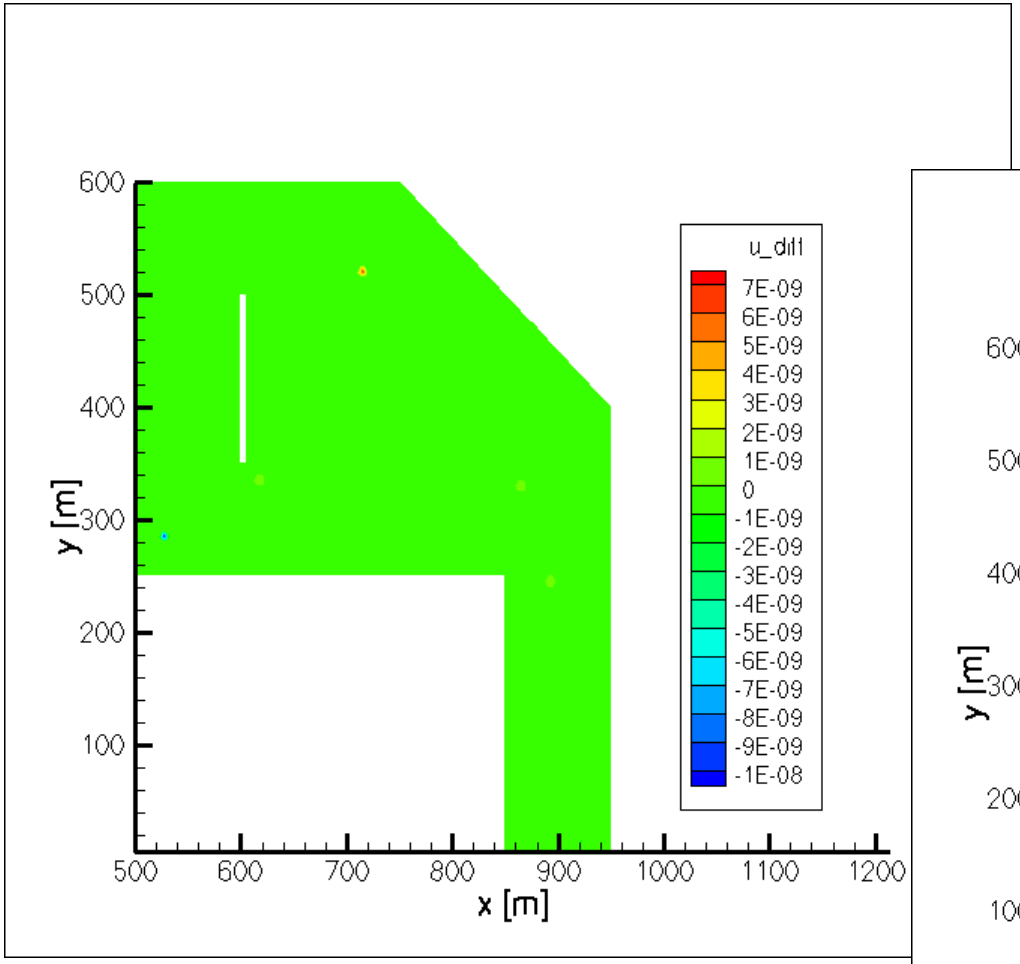
Harbour

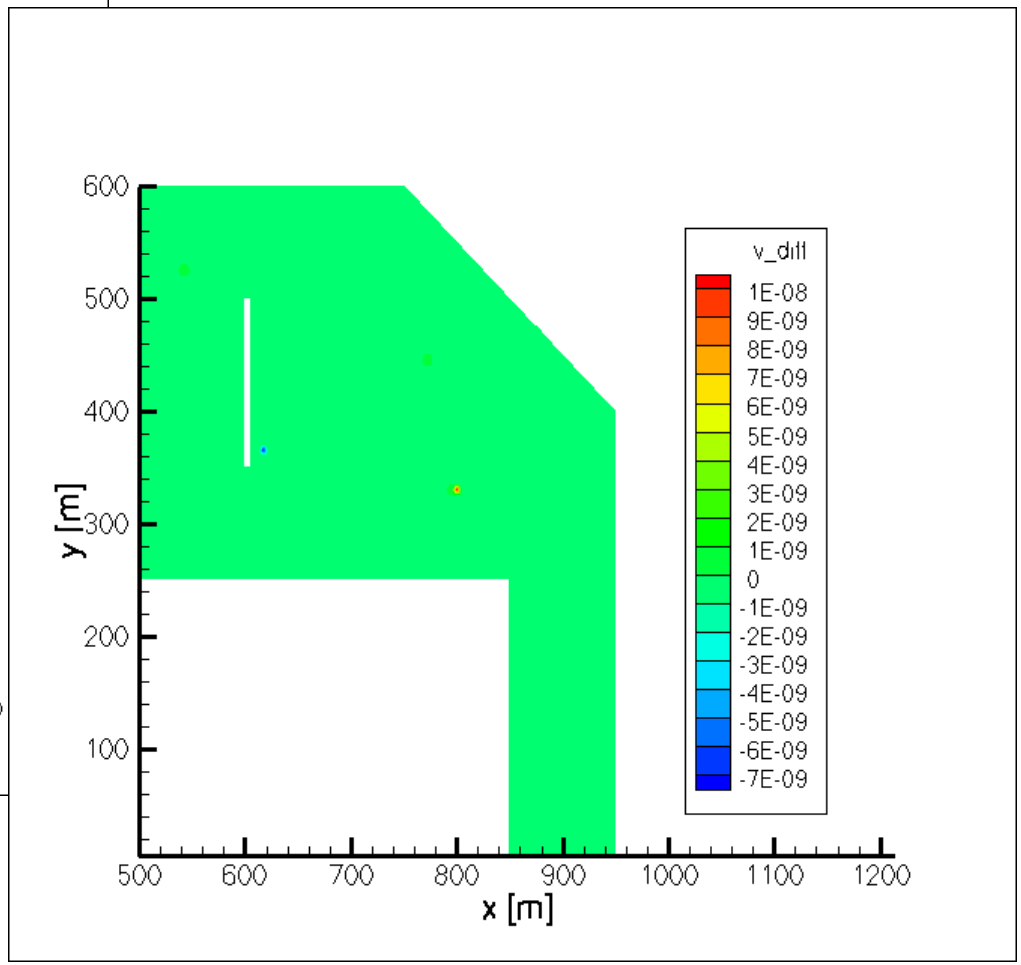




A regular mesh of triangles, flat bottom, short waves

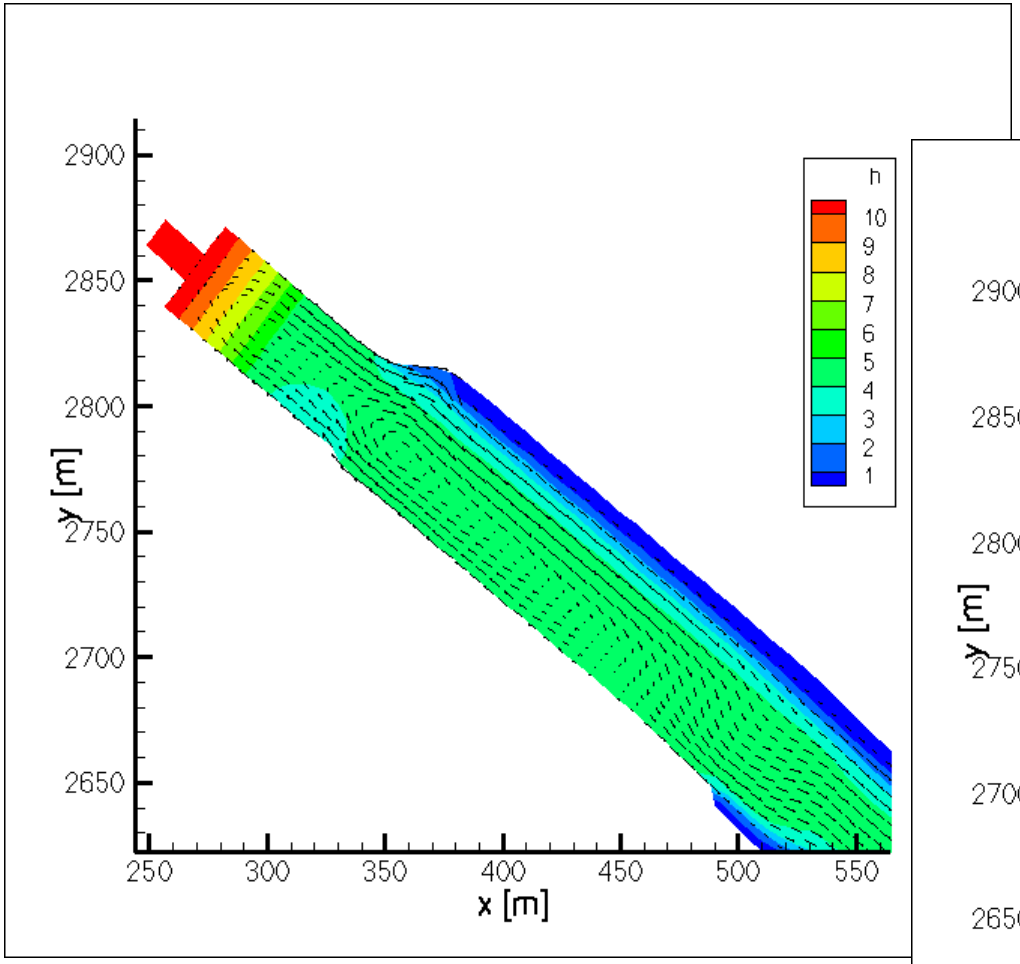
Harbour

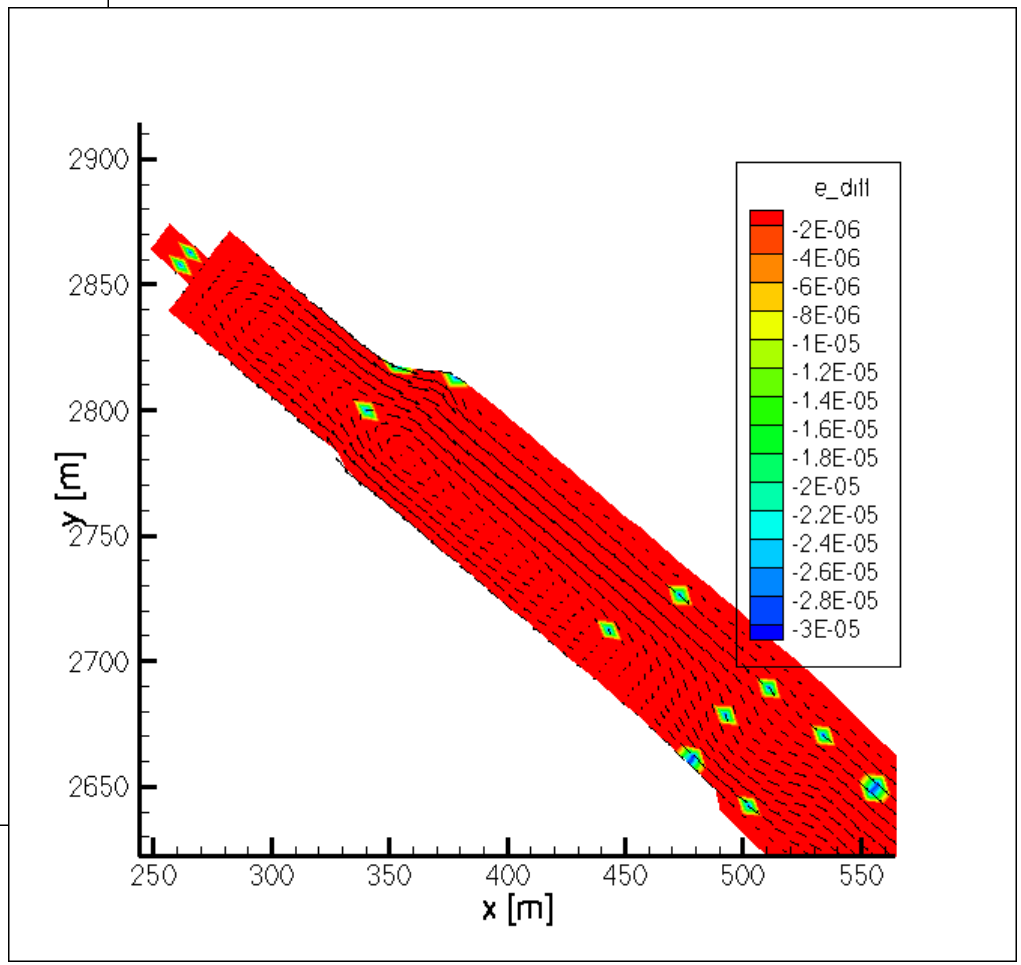




A completely non-hydrostatic test case

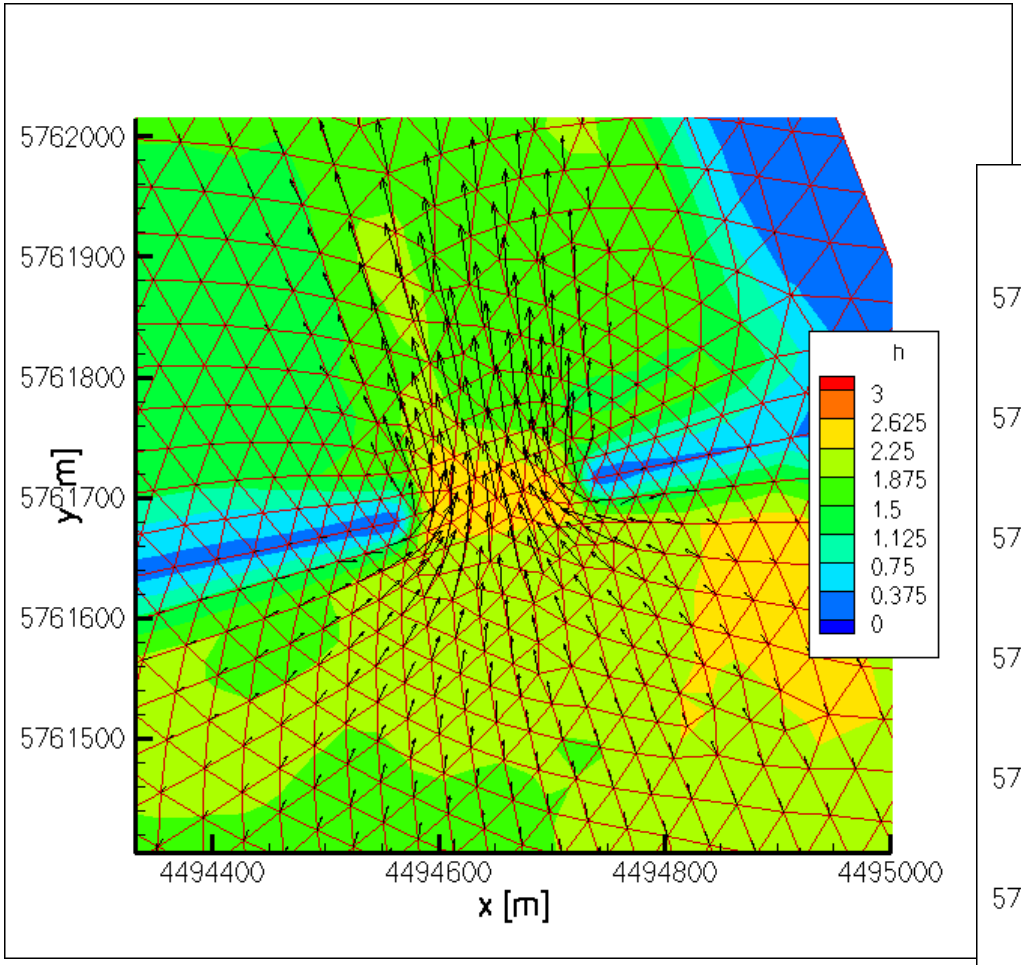
Lock waves

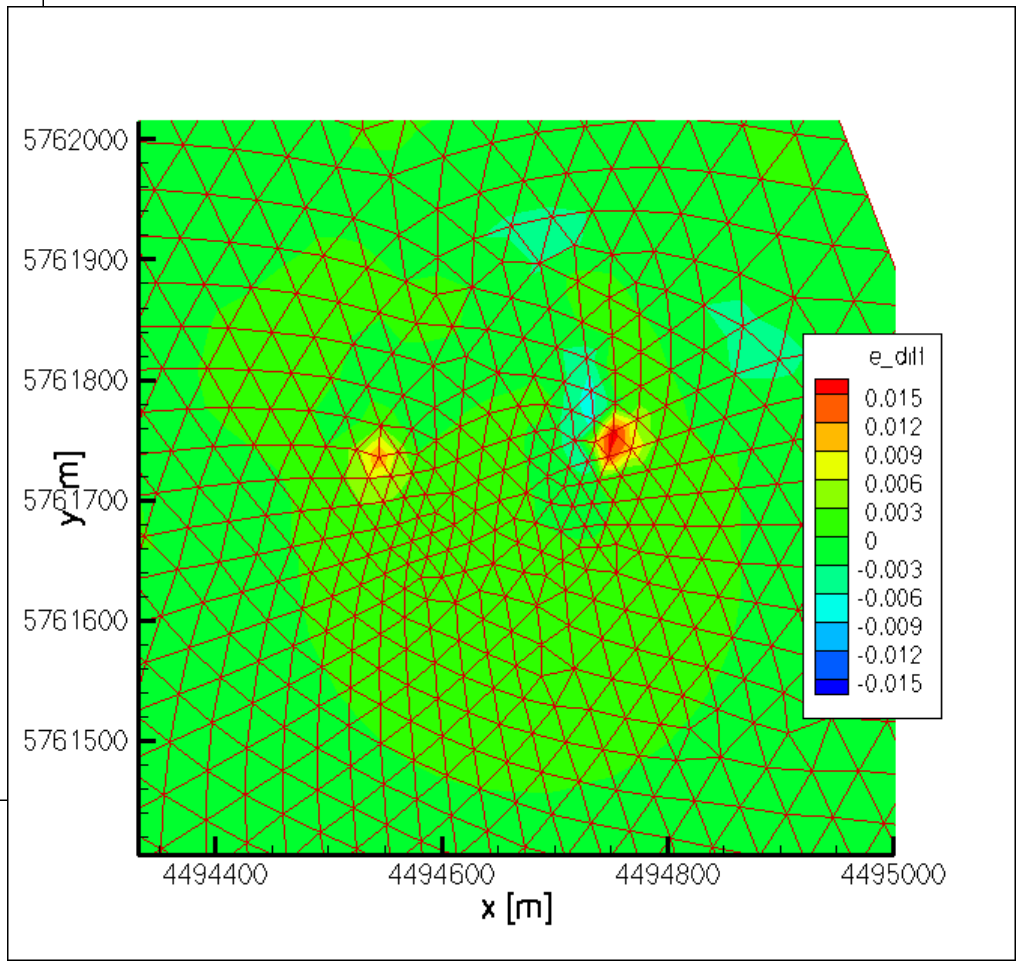




A very regular mesh, quadrangles

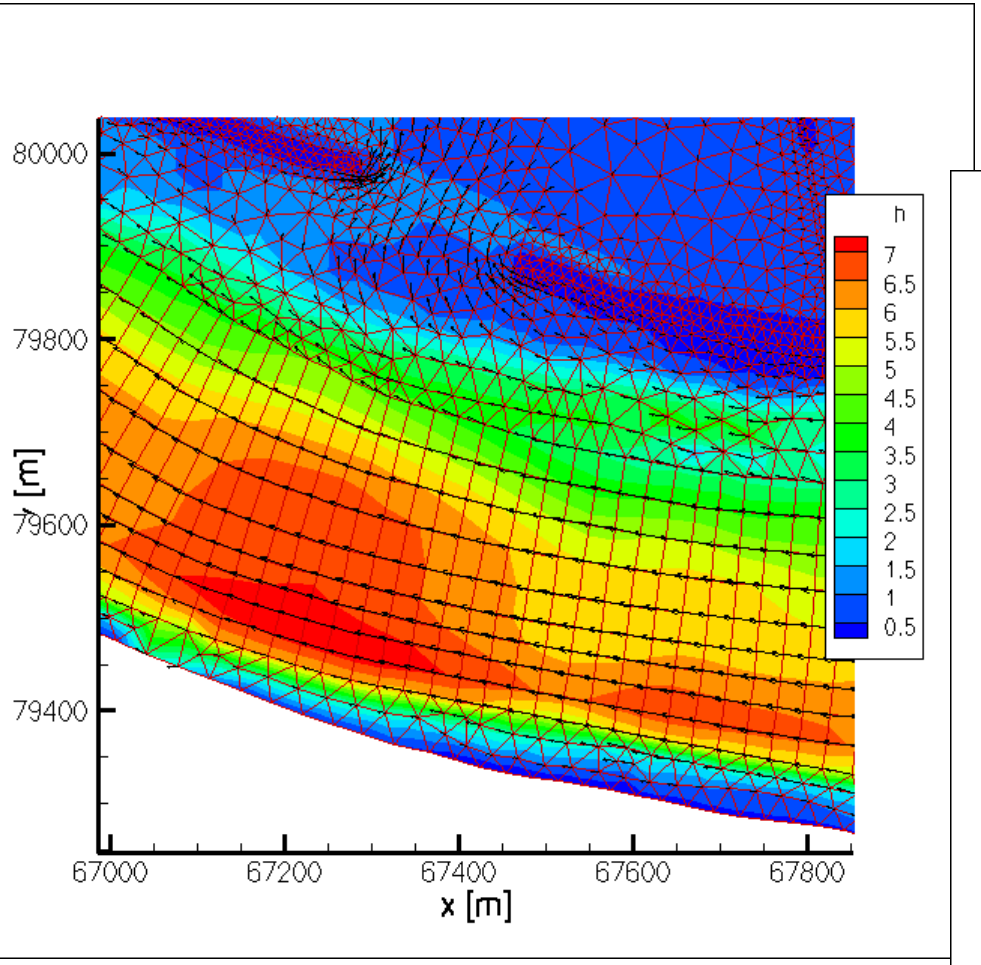
Saale/Elbe

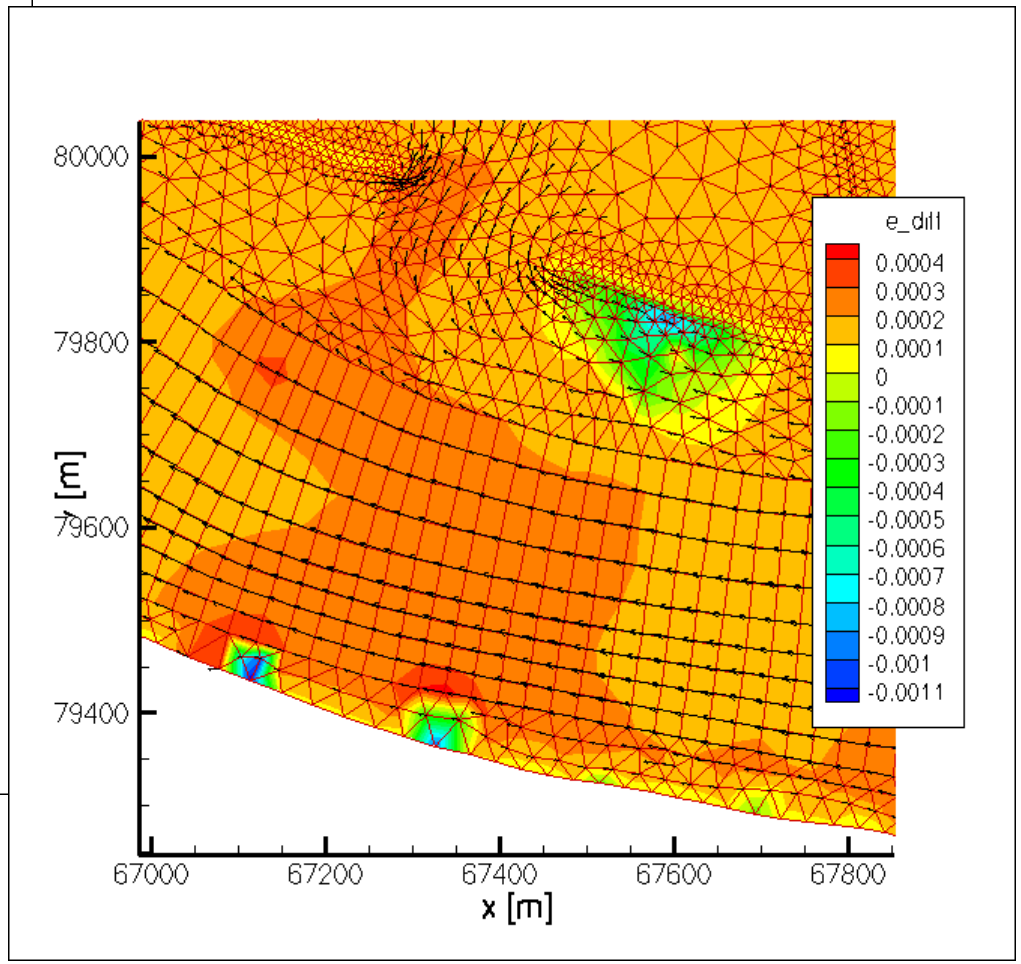




dry/wet boundaries,
steep bottom gradients

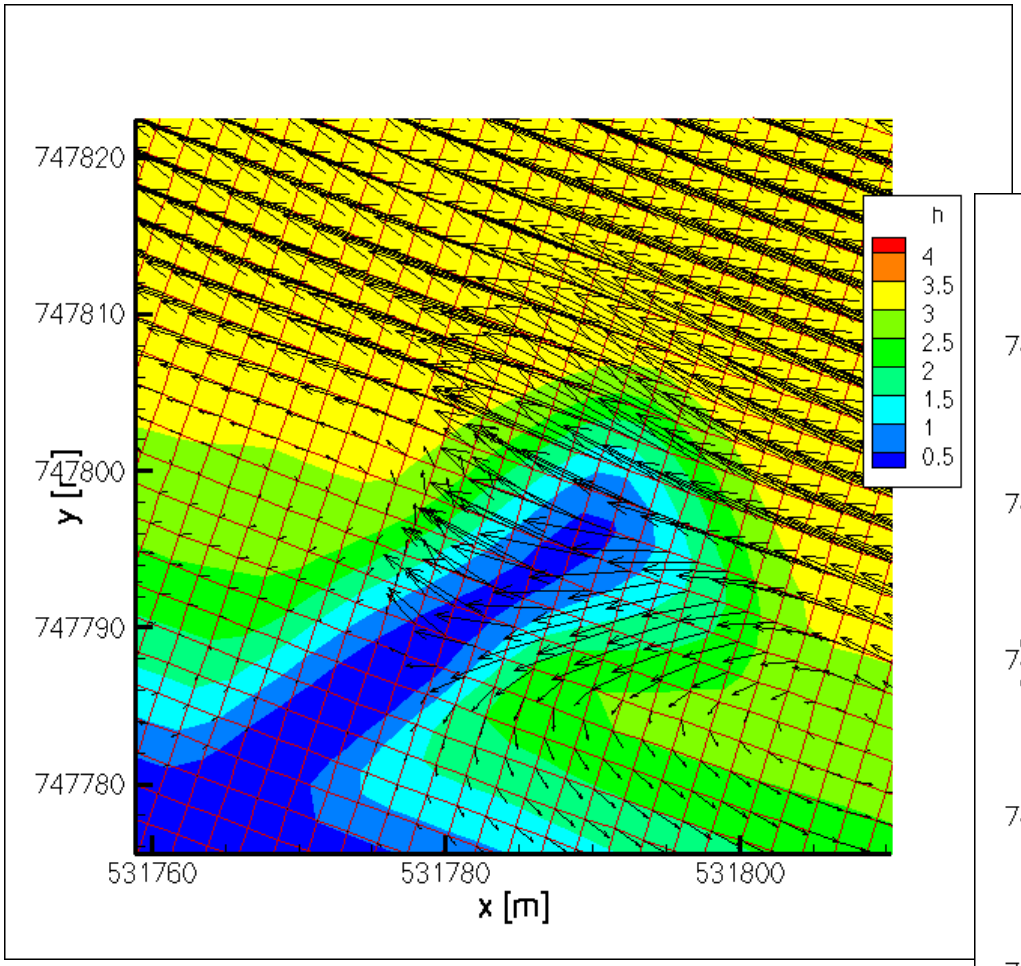
Lenzen

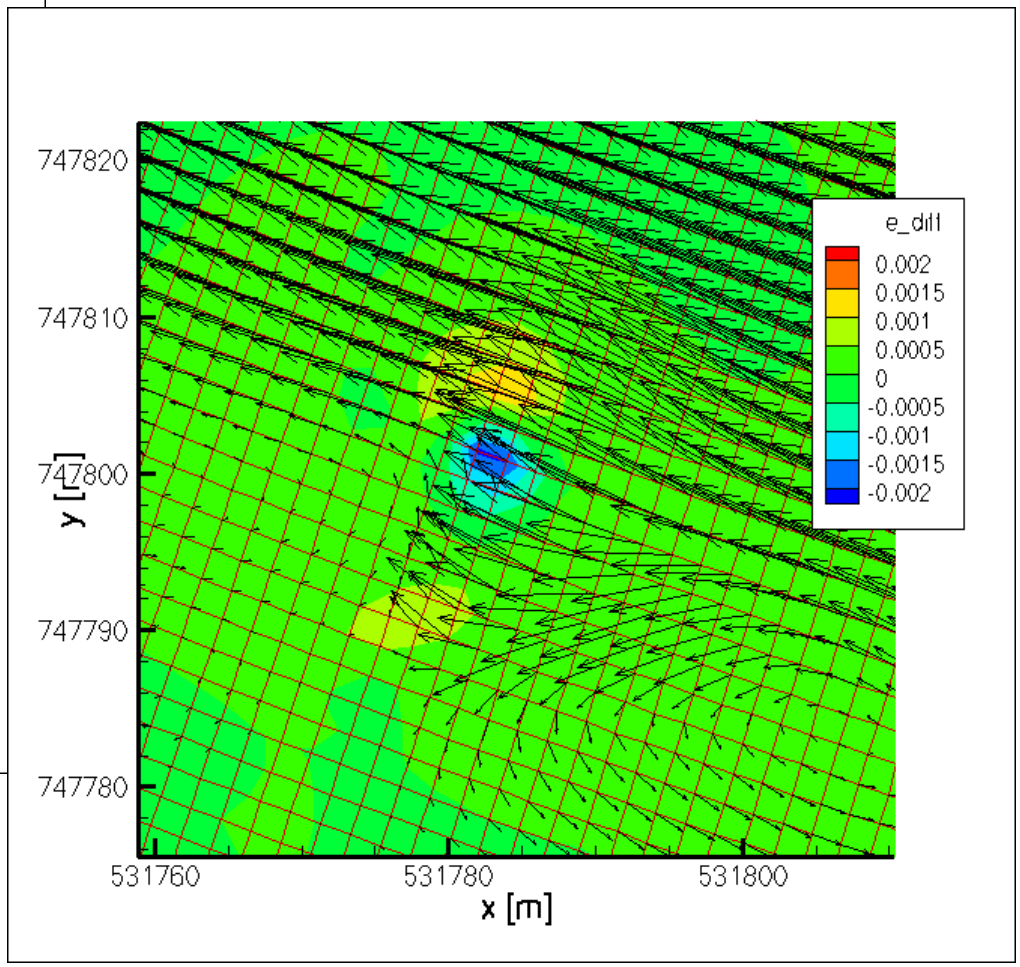




dry/wet boundaries,
steep bottom gradients

Coswig

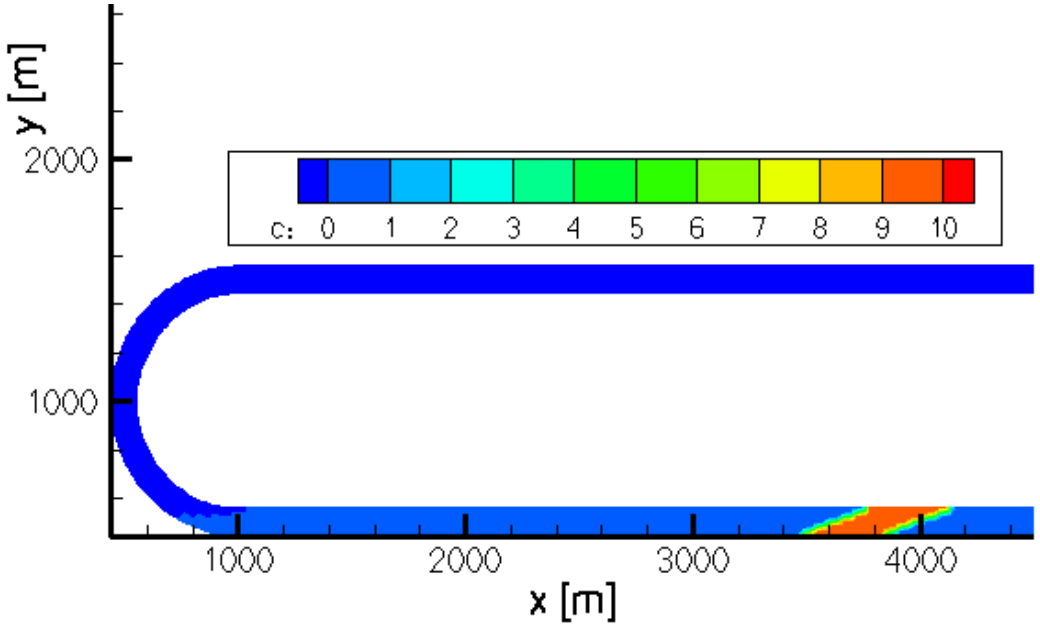


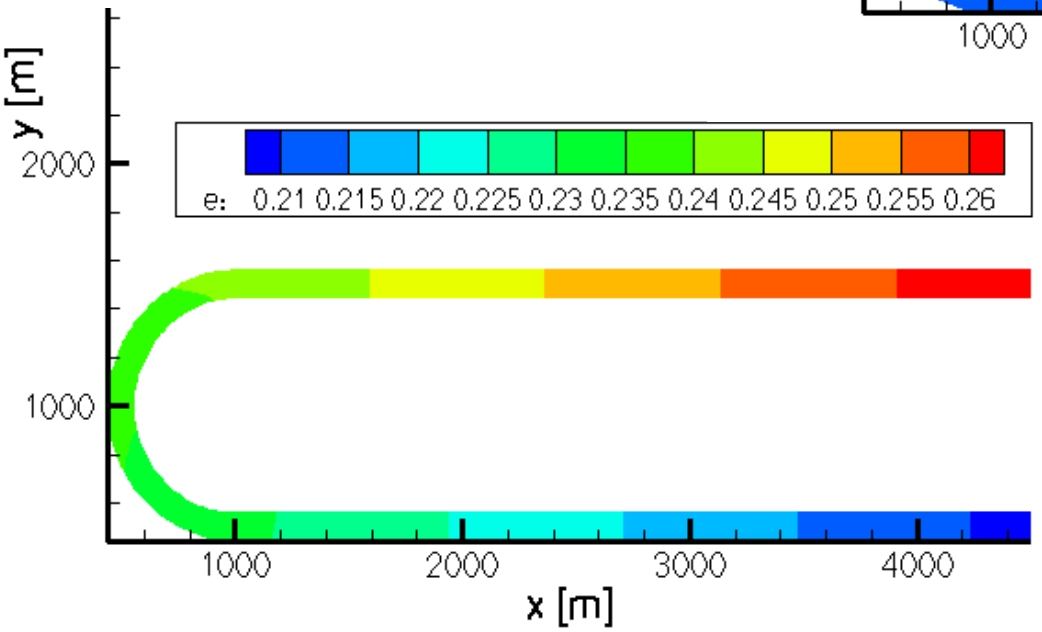


just overflown groynes

U-bend, 2D

A concentration signal of C=10 moves in the 2D U-bend channel current

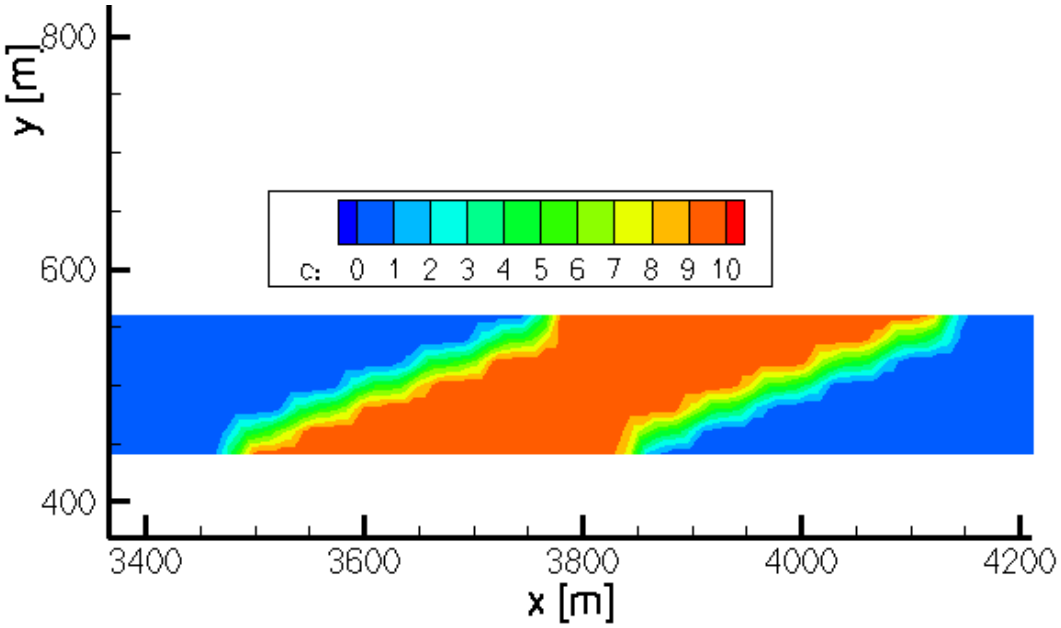


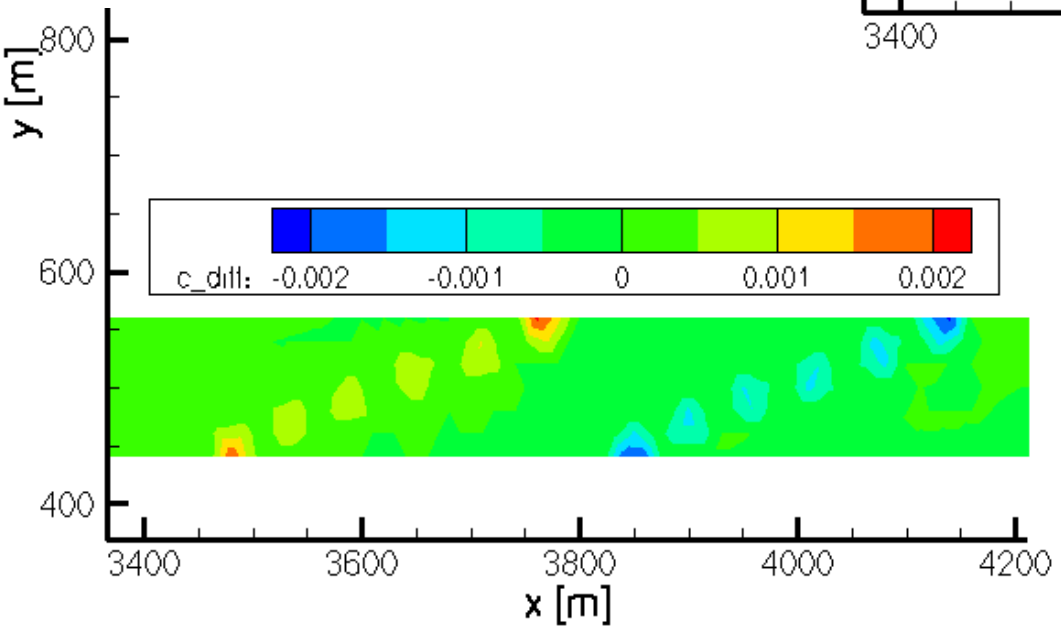


Note the sharp gradients of the concentration distribution

C_diff, 2D

Note the sharp concentration differences

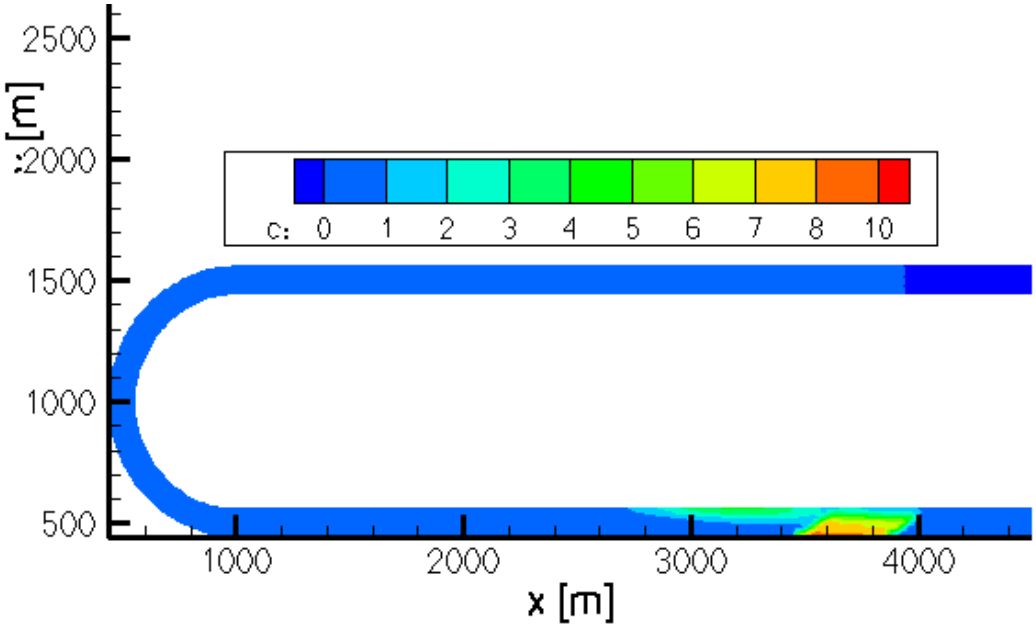


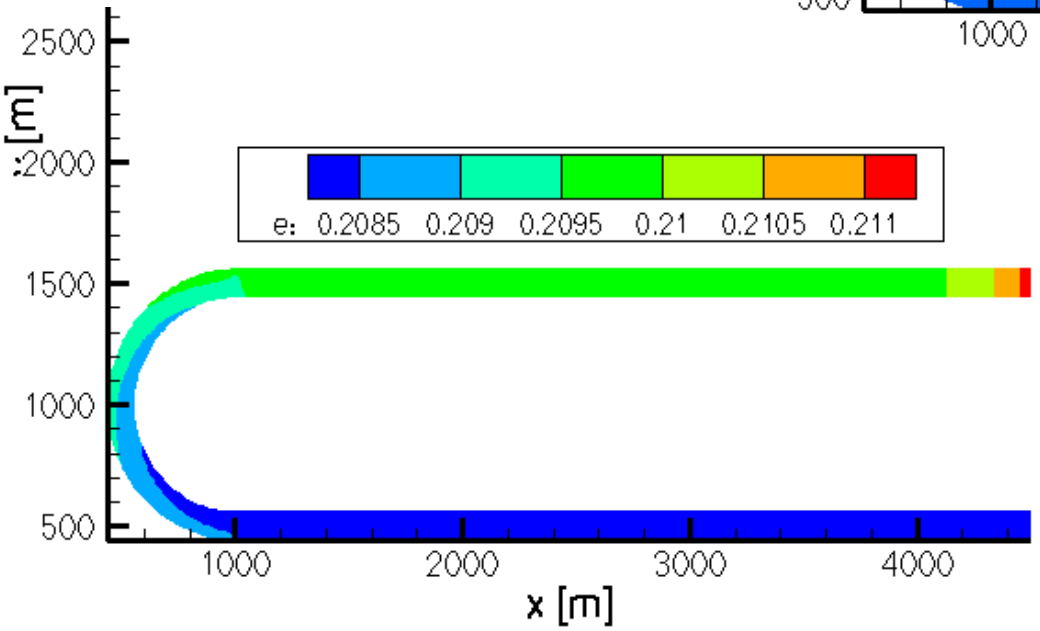


The concentration signal moves slightly too slowly in the parallel case (?)

U-bend, 3D

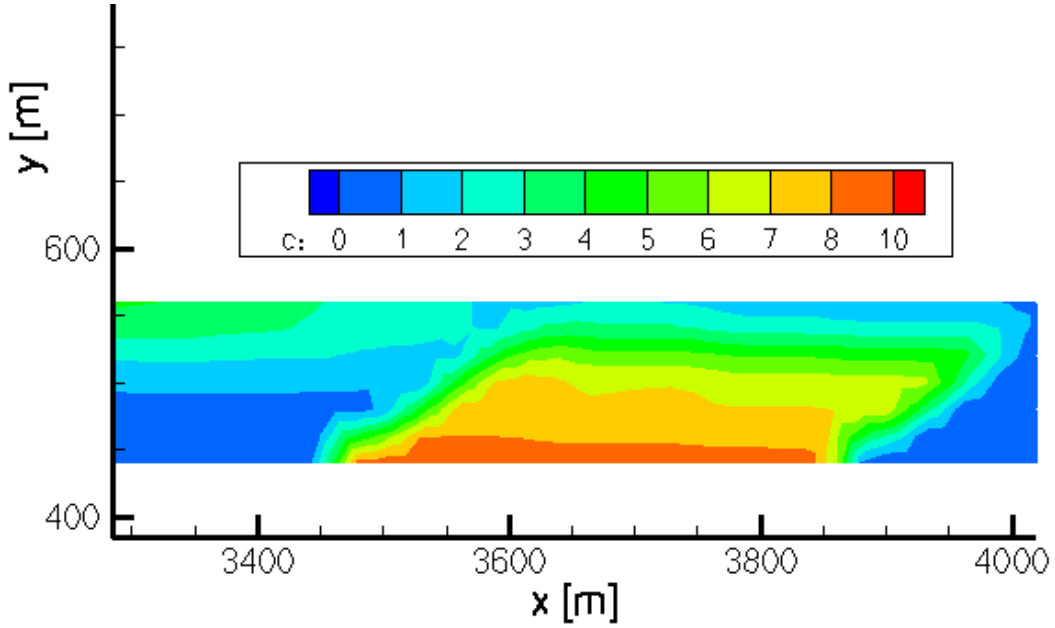
A concentration signal of C=10 moves in the 3D U-bend channel current



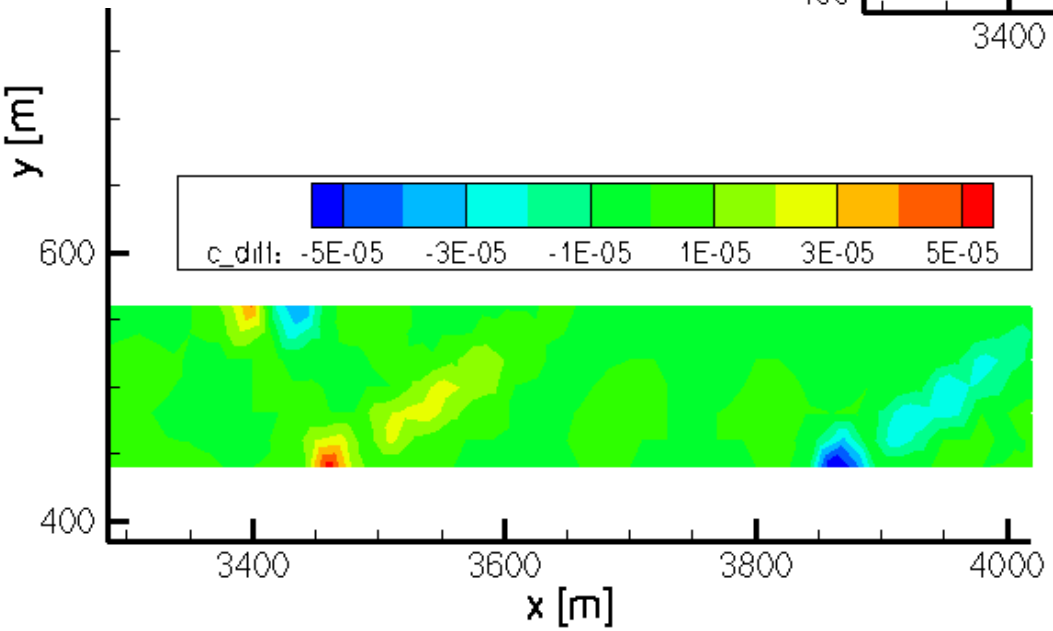


Note the free surface and concentration distribution are much different than in the 2D case

C_diff, 3D

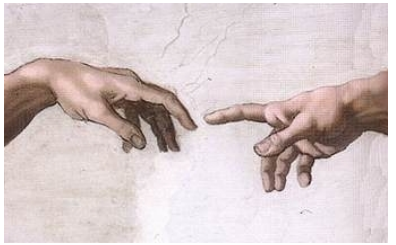


Note: this is the vertically integrated concentration profile



The concentration signal moves slightly too slowly in the parallel case (?)

Recommendations



In order to diminish the differences between serial and parallel runs:

- Avoid any aspects of the mesh dependency
- Do not simplify the physics
- Transport phenomena – small errors add up...

Scalability

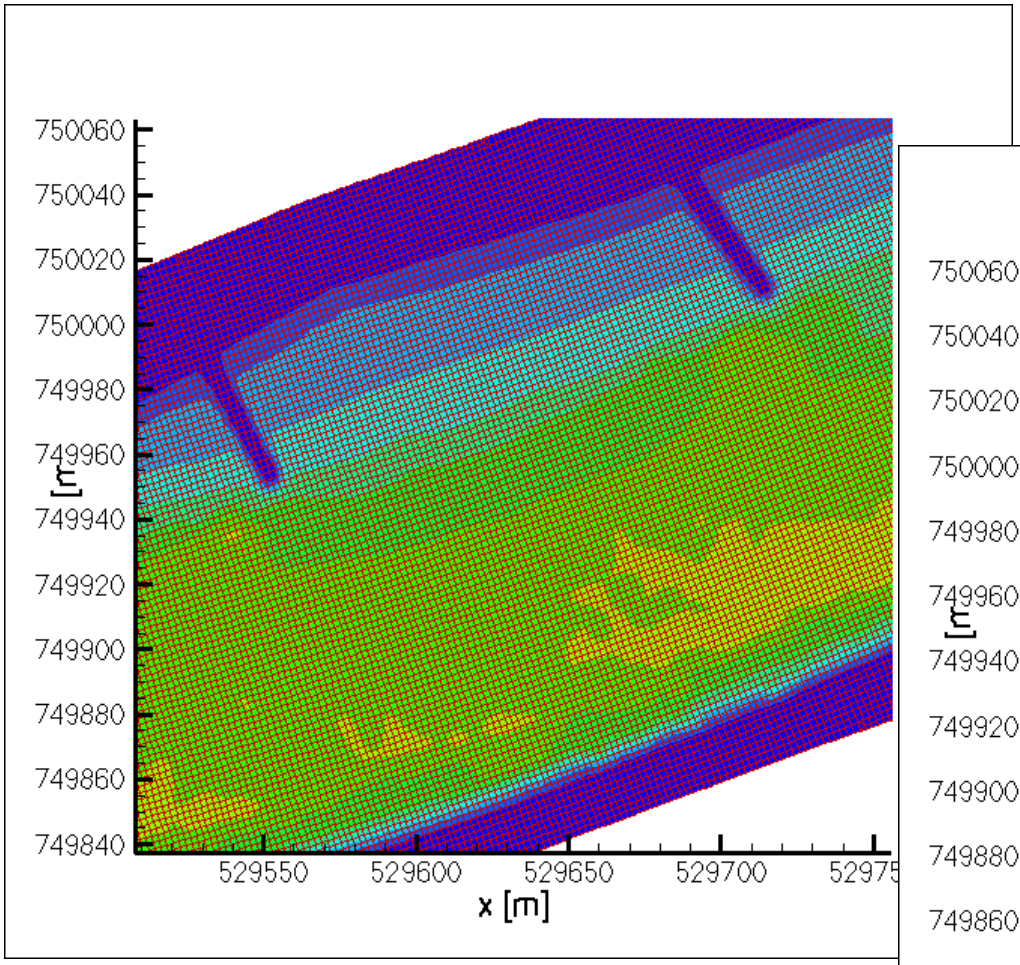
Speedups obtained with the MPI-parallel UnTRIM

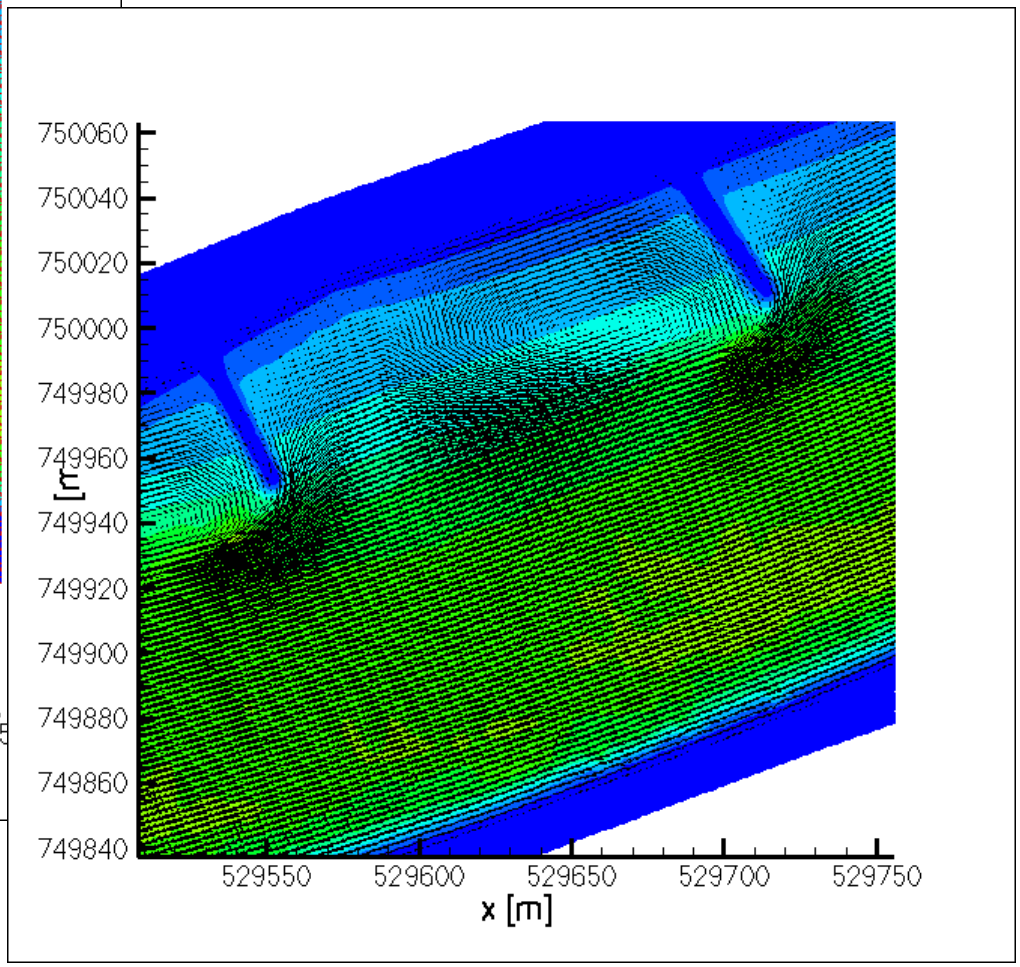
obelix.karlsruhe.baw.de

- **obelix+idefix (**4+1 cabinets)
- SGI Altix 3600
- **256+48** 1600 MHz Itanium-2 processors, 6MB cache
- 256+48GB **shared** memory
- 64-bit SuSE Linux 10.0
- CPU-Sets with PBS-Pro
- Intel and Gnu compilers
- OpenMP and MPI

- **A *state-of-the-art* parallel computer**

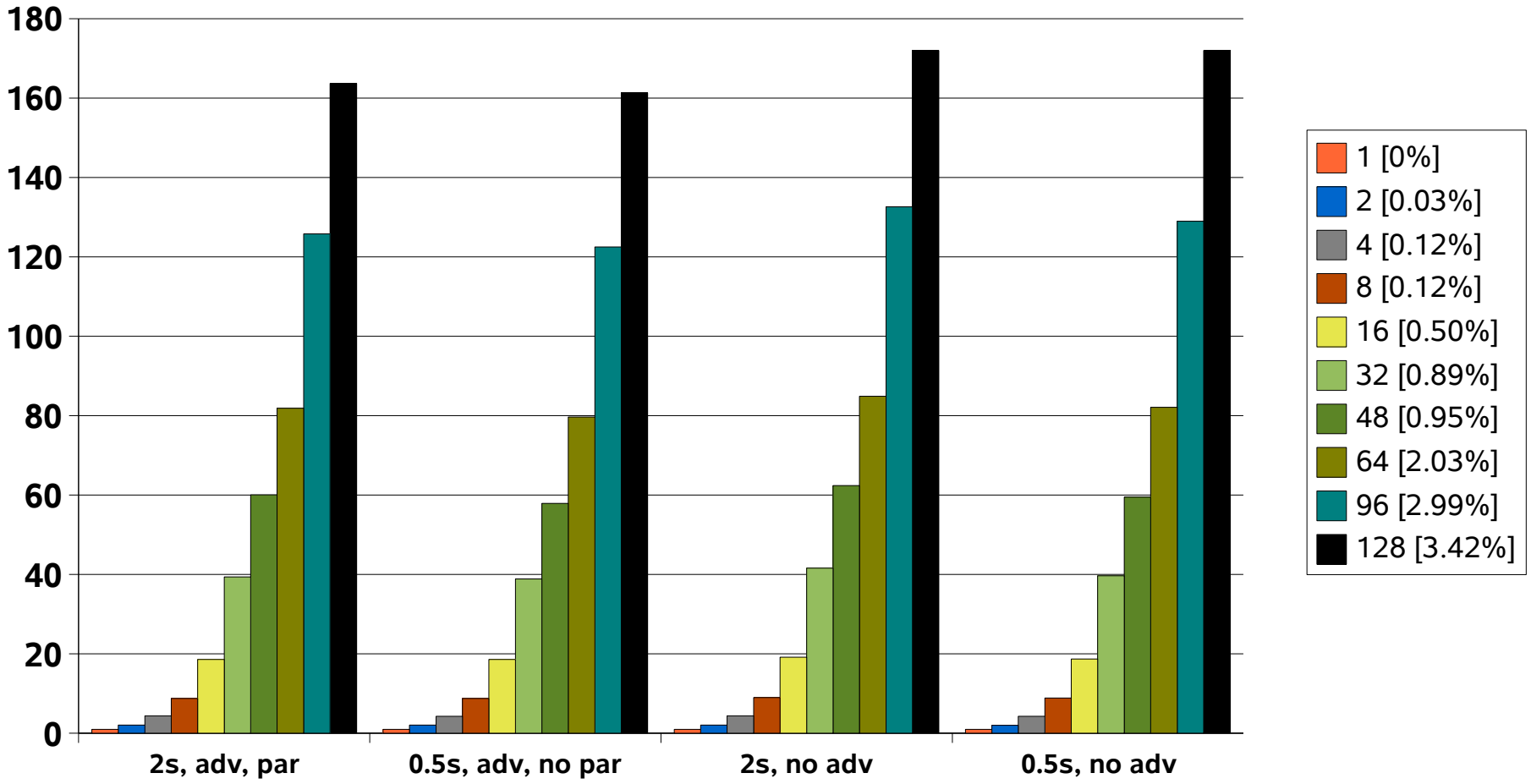
Coswig



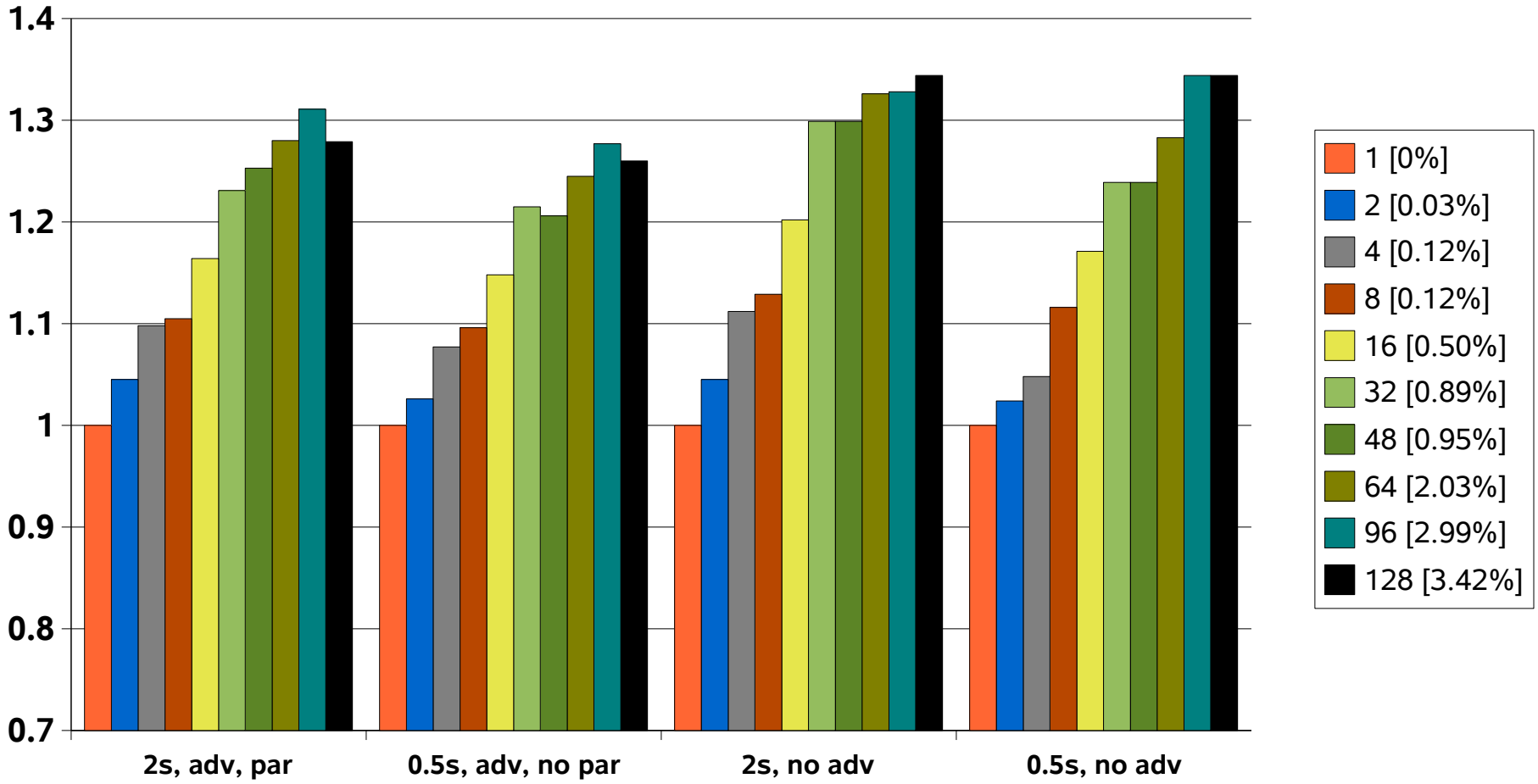


14km river stretch - a mesh of very regular quadrangles with dx=2m, dz=0.5m, ne=738485, nk=22, n3e = 8006838

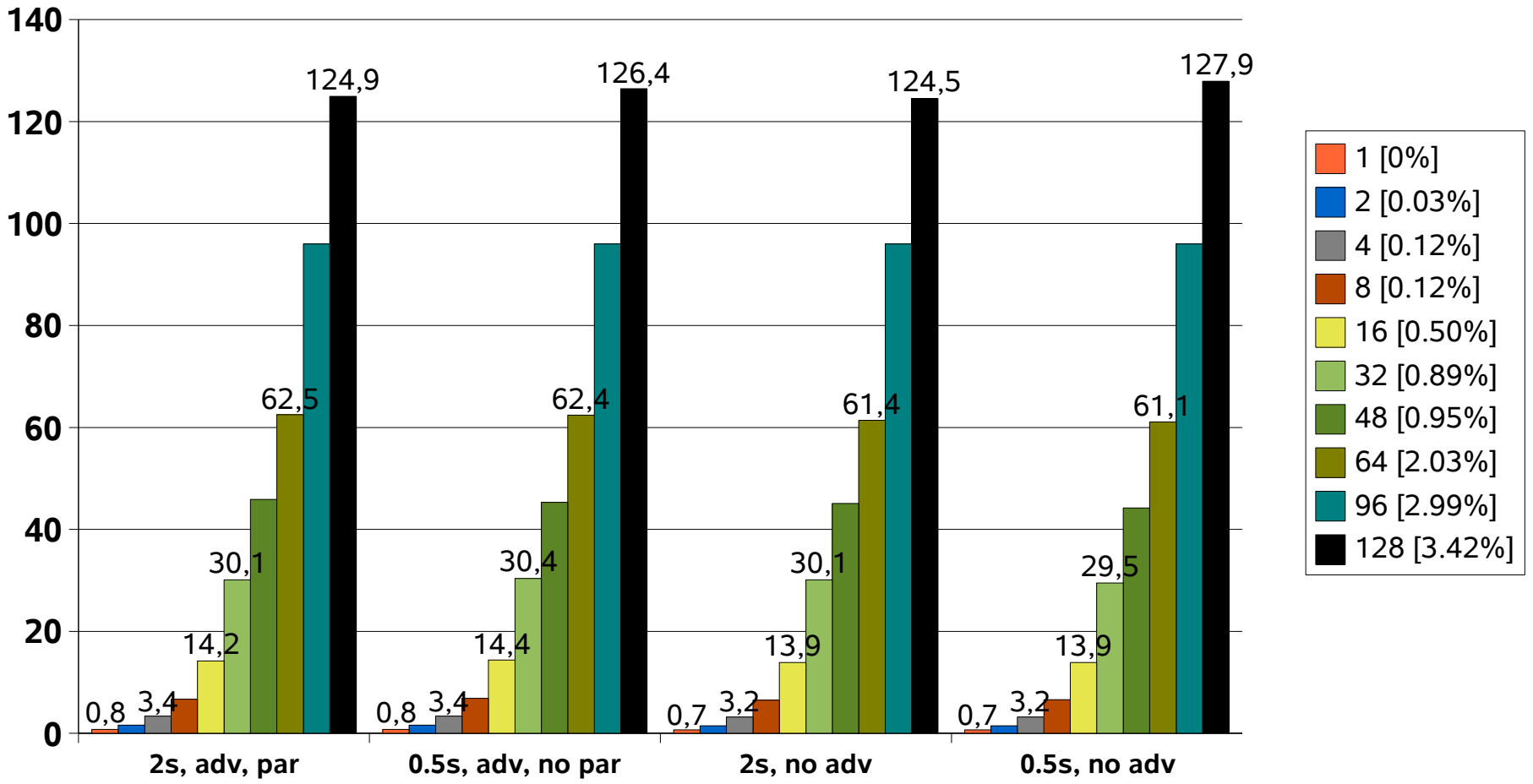
Coswig - speedup relative to 1p. (middle water, 2D, ne=738485)



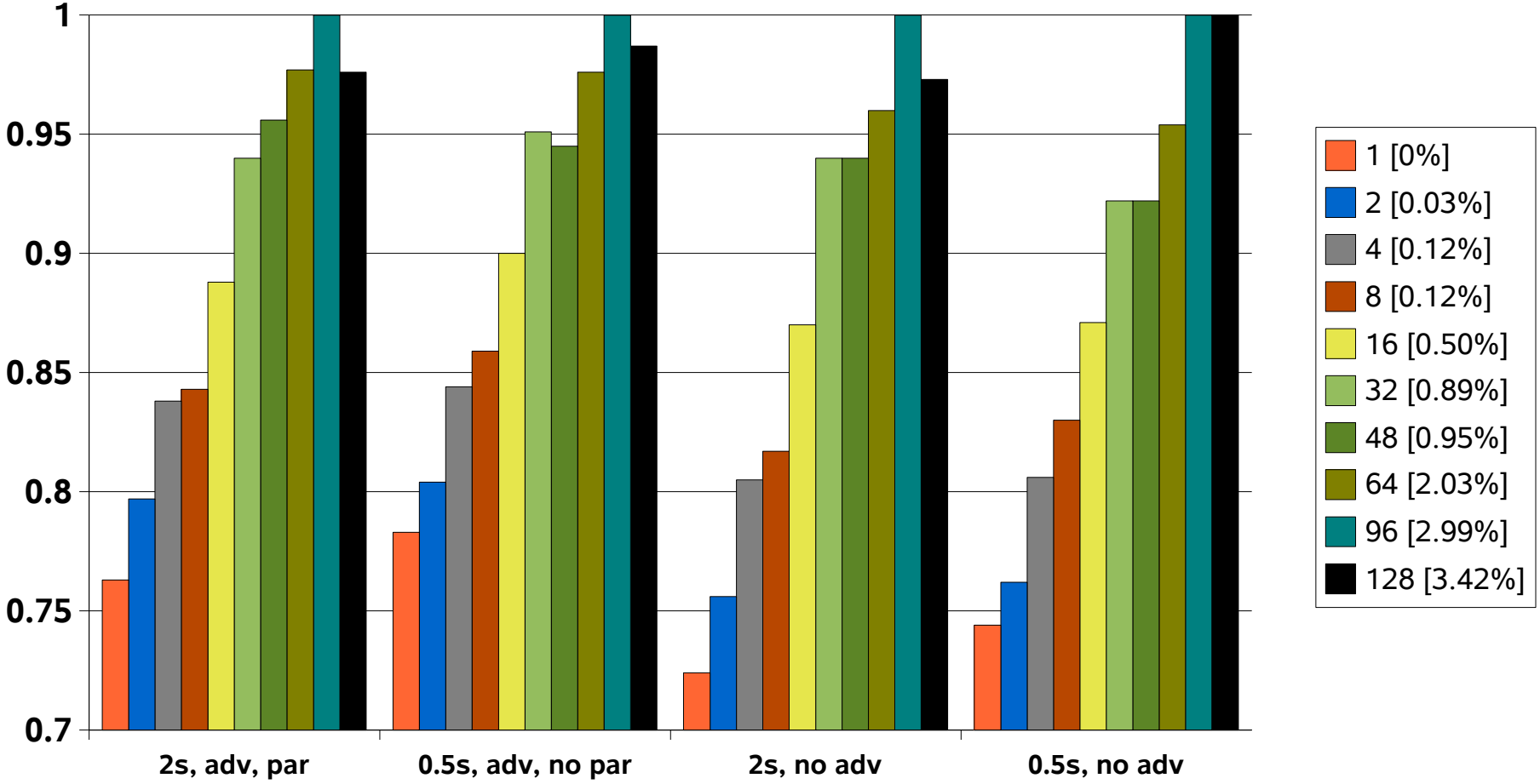
Coswig - efficiency relative to 1p. (middle water, 2D, ne=738485)



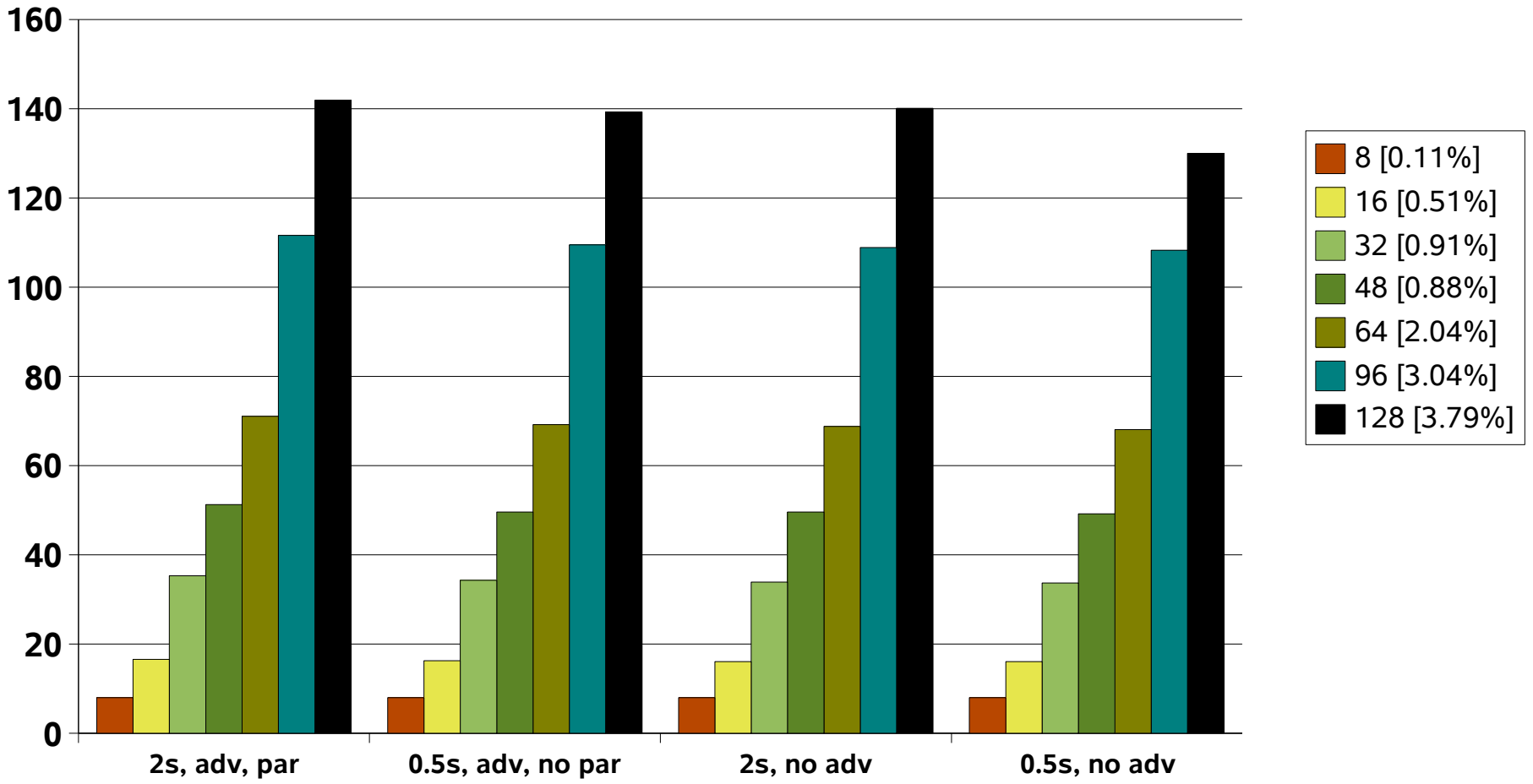
Coswig - speedup relative to 96p. (middle water, 2D, ne=738485)



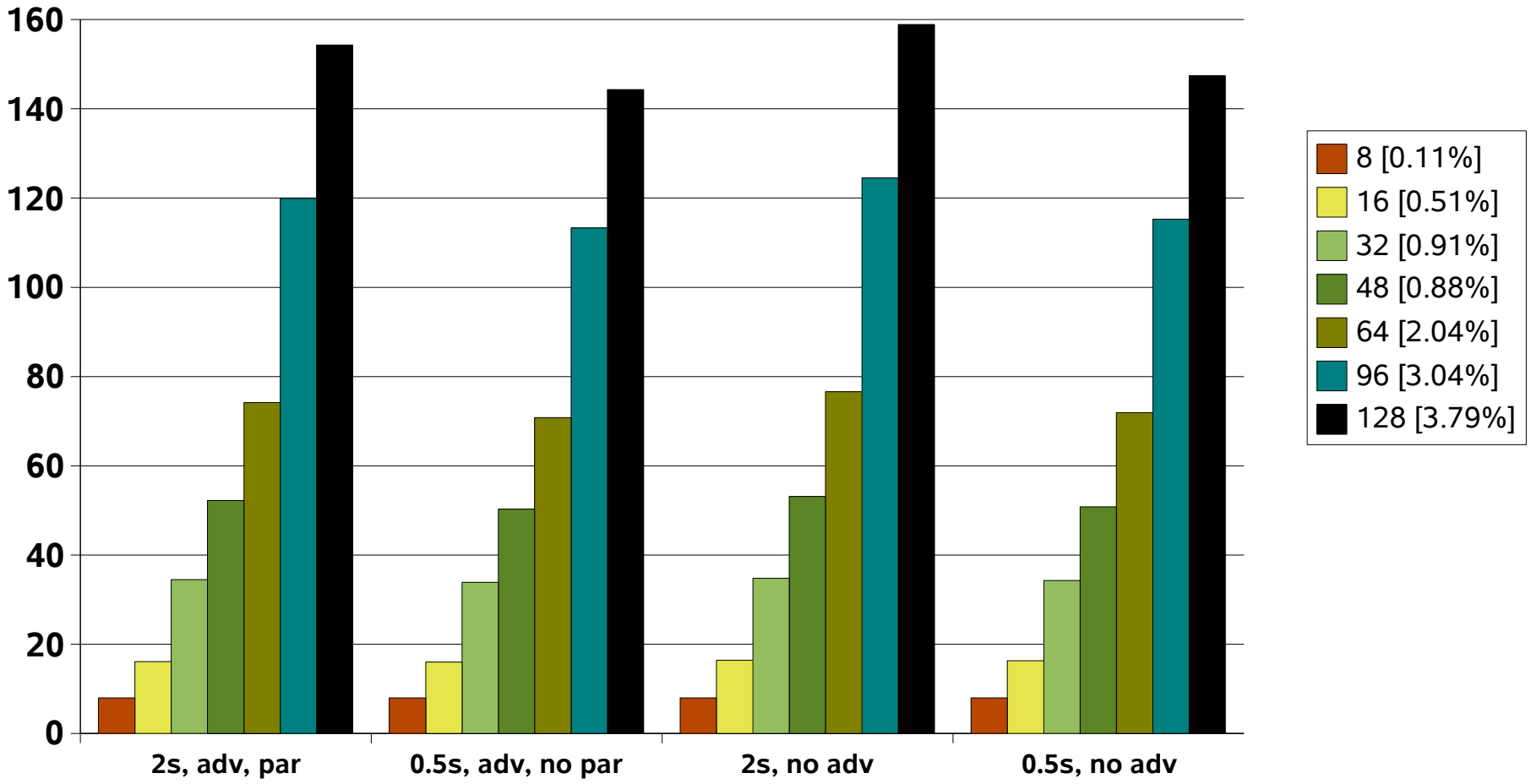
Coswig - efficiency relative to 96p. (middle water, 2D, ne=738485)



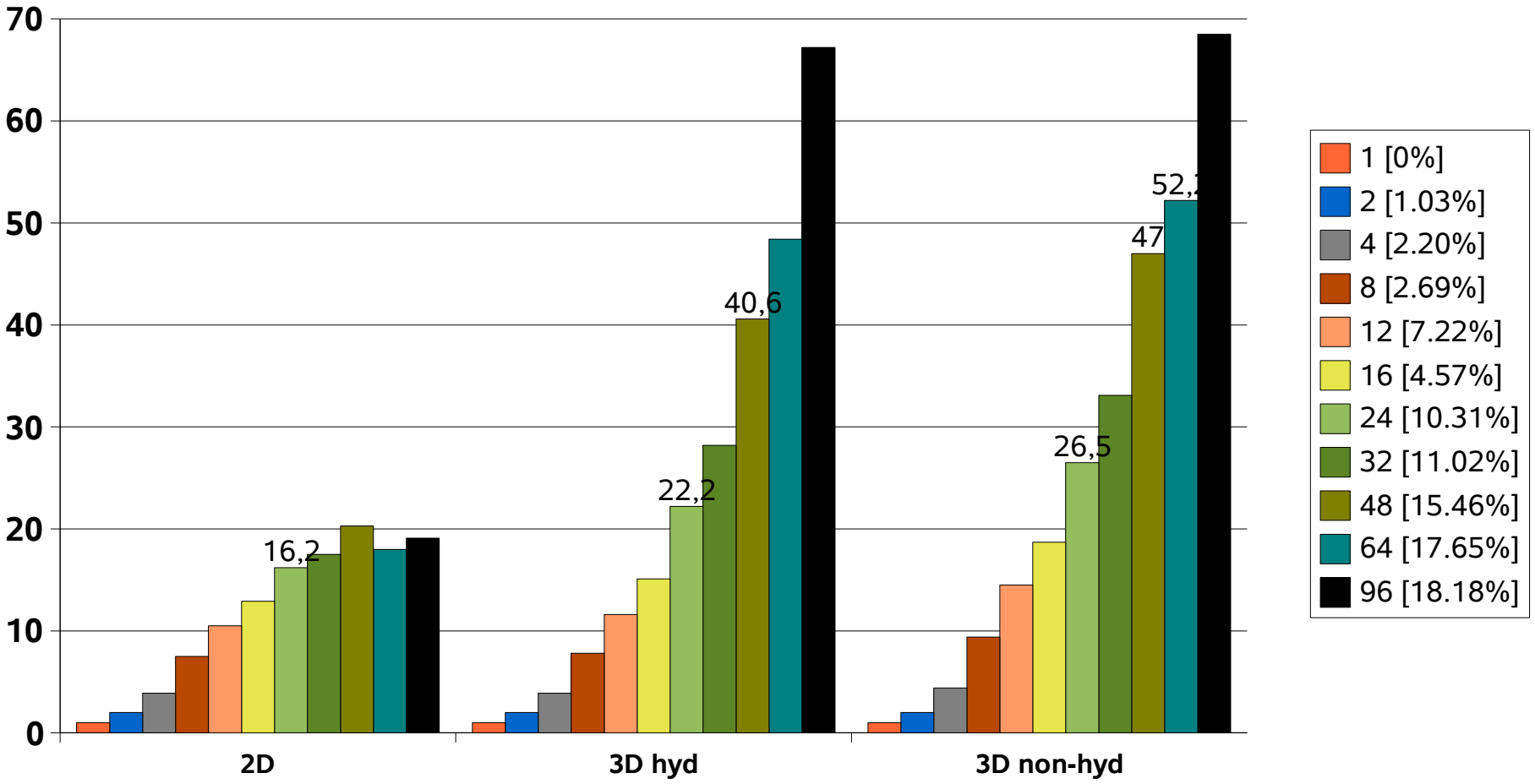
Coswig - speedup relative to 8p. (middle water, 3D hyd, n3e=8006838)



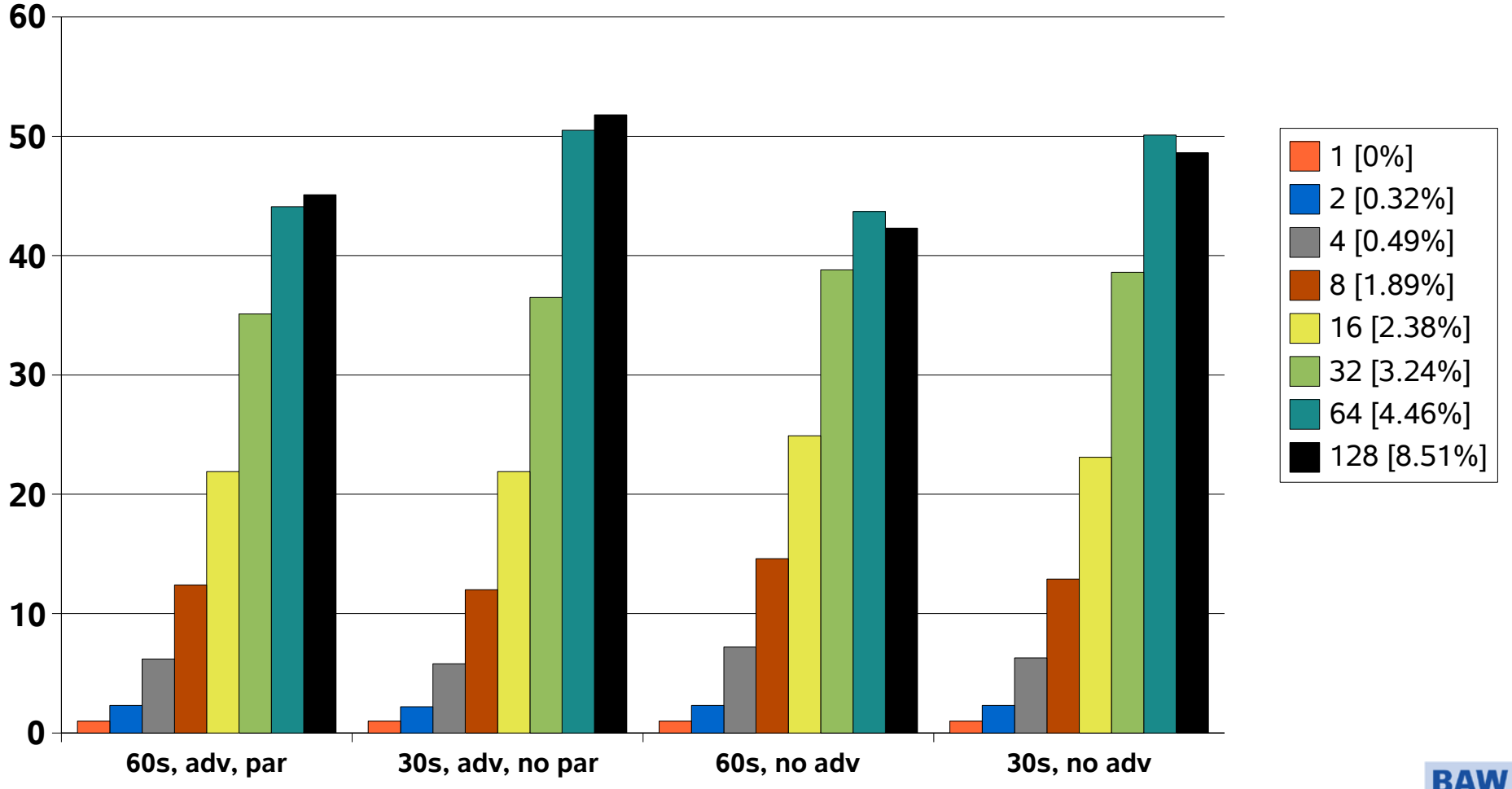
Coswig - speedup relative to 8p.
(middle water, 3D, non-hyd, n3e=8006838)



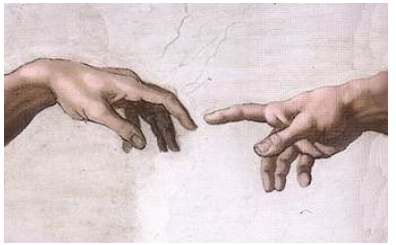
Harbour (ne=13069, n3e=405139)



The Saale/Elbe confluence (ne=126402, dx=15-20m)



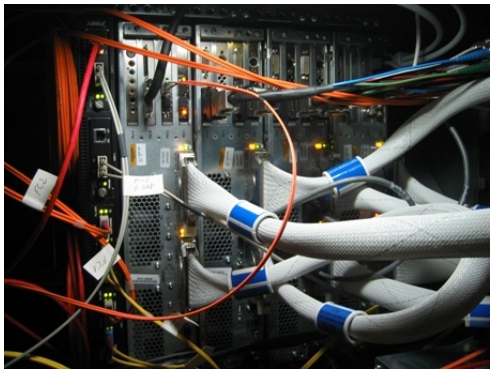
Recommendations



In order to maintain a good scalability of the scheme, avoid:

- runs with more than ca. 10% halo cells
- partitioning “in vain” - e.g. of dry-only cell areas
- large numbers of iterations in equation solvers, diffusion and transport schemes (>50) and large number of traceback interface crossings (>5)

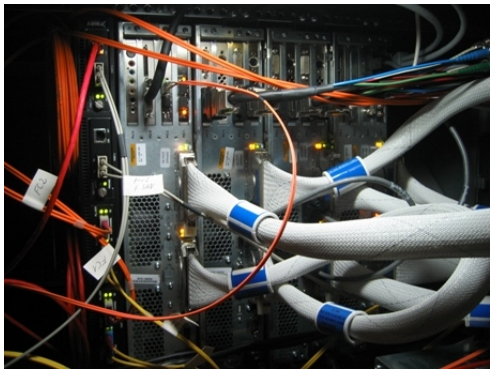
Reached



- A parallel UnTRIM implementation without compromising the properties of the serial code

- A good scalability due to:
  - communication adequately designed for the significant parts of the algorithm
  - minimal amount of data exchanged between processors

To be addressed (soon?)



- Portability of the parallel advection scheme code
- Performance measurements (professional tools)
- Point-to-point communication with MPI data types
- Parallel User Interface (boundaries...)
- Parallel I/O (for profiles, time series, discharges...)
- Partitioning methods (boundaries...)
- *Cache misses – improve the 'data locality'*

I listen to all questions!

Additional transparencies

...for discussions, etc.

Discussion

Existing and alternative advection schemes applicable for UnTRIM

Lagrangian advection scheme

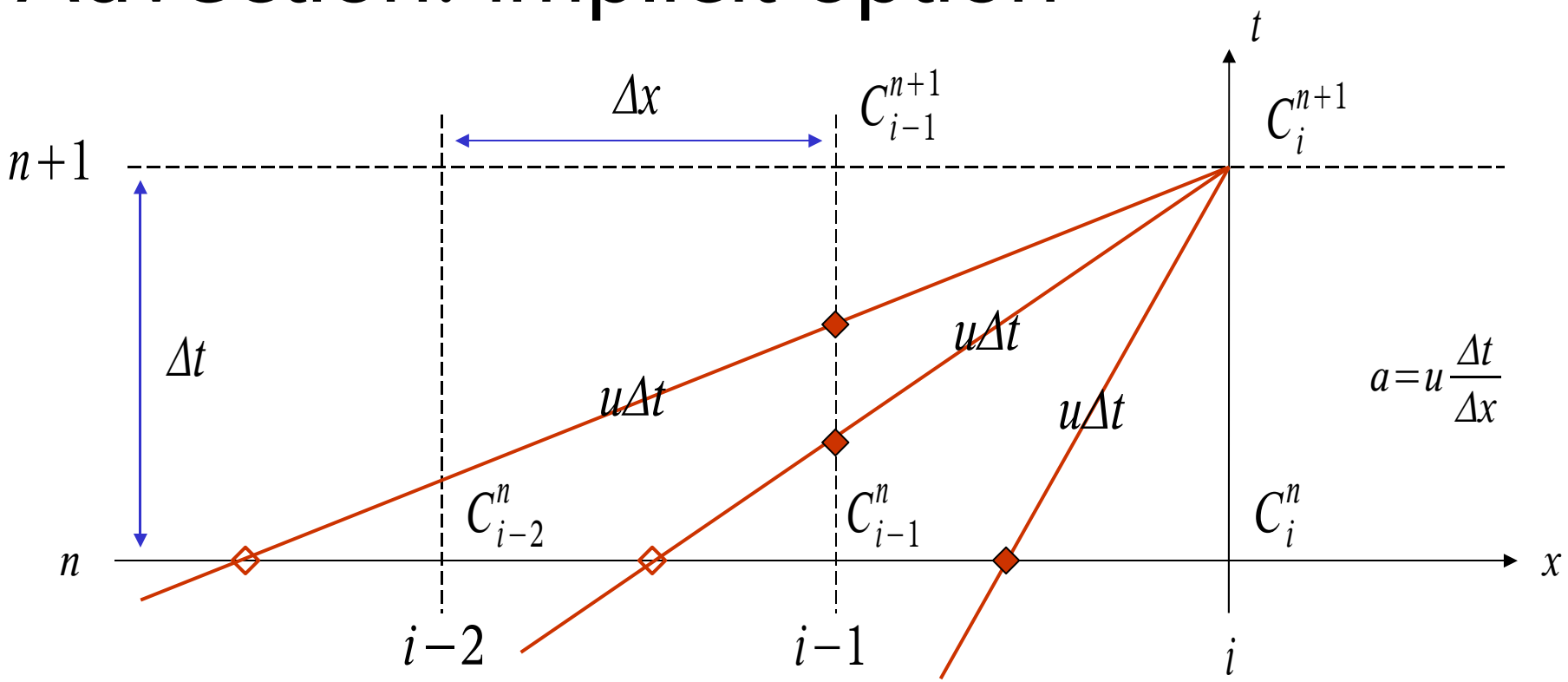
- Contains of three parts:
  - algorithm for the streamline tracking backward in time over the mesh
  - interpolation scheme for the values at the characteristic curve foot
  - applying the obtained values
- Unconditionally stable, but…
- …the quality depends…
  - mostly on the interpolation scheme (linear, bilinear,…)
  - less on the tracking algorithm by high-quality meshes

Advection: an implicit scheme?

- Advantages:
  - a linear equation system is to be solved with simple methods
  - straightforward to implement in parallel
  - “implemented successfully in one of the older programs of TU Delft” [G.Stelling]

- Disadvantages:
  - implemented for linear interpolations, properties for higher orders and unstructured meshes?
  - “an implementation can be considered only when the existing algorithm is very inefficient in parallel” [V.Casulli]

Advection: implicit option



$$C_i^{n+1} = C_{i-a}^{n} - u\frac{\Delta t}{\Delta x}\left(C_i^{n} - C_{i-1}^{n(+1)}\right)$$

Advection: Eulerian scheme?

- Implement a modern, conservative, etc. Eulerian schemes):
  - very simple to implement in parallel
  - unfortunately: the Courant-Number limitation is valid

- SUNTANS (*Fringer, Gerritsen, Street, 2006*) uses such scheme (by *Perot 2000)* in parallel:
  - tested for deep waters (internal waves)
  - unstable by tidal flats (but maybe OK for 2D)
  - Lagrange scheme newly implemented [O.B.Fringer, pers.comm.]
  - well known properties only for structured meshes [V.Casulli]

Extending overlapping halos

- Advantages:
  - avoiding any changes of the original code
  - only domain decomposition and communication software

- Disadvantages:
  - the overlapping width cannot be foreseen easily
  - for larger Courant numbers always too many halo cells
  - for larger number of processors always too much communication, what affects the scalability strongly [O.B.Fringer, pers. comm.]

Conclusions

All existing and alternative advection schemes applicable for UnTRIM can be treated in the framework of the developed MPI library:

- Lagrange schemes: autonomous tracking with global communication

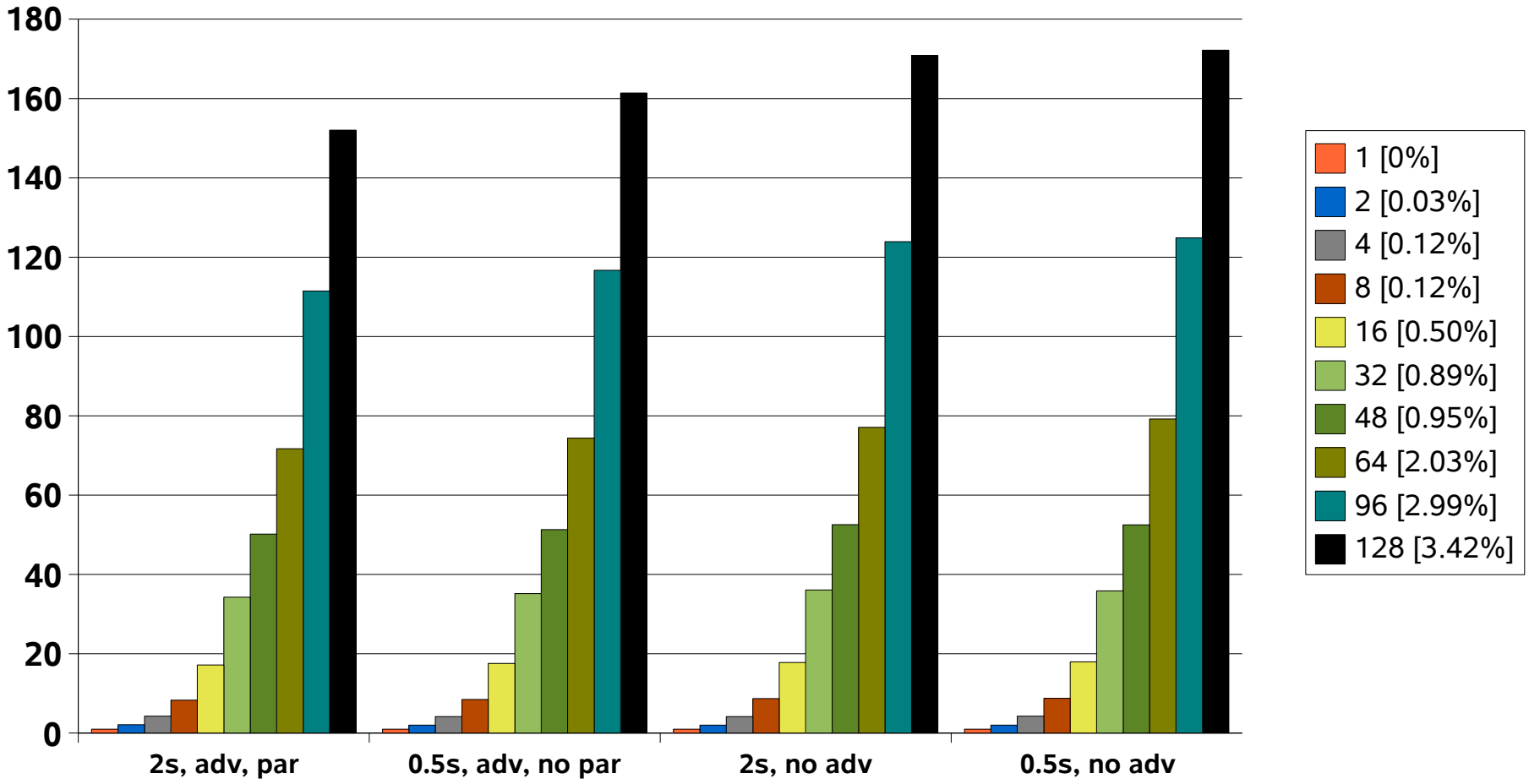
- Eulerian & implicit schemes – halo swapping with point-to-point communication

Advection speedup

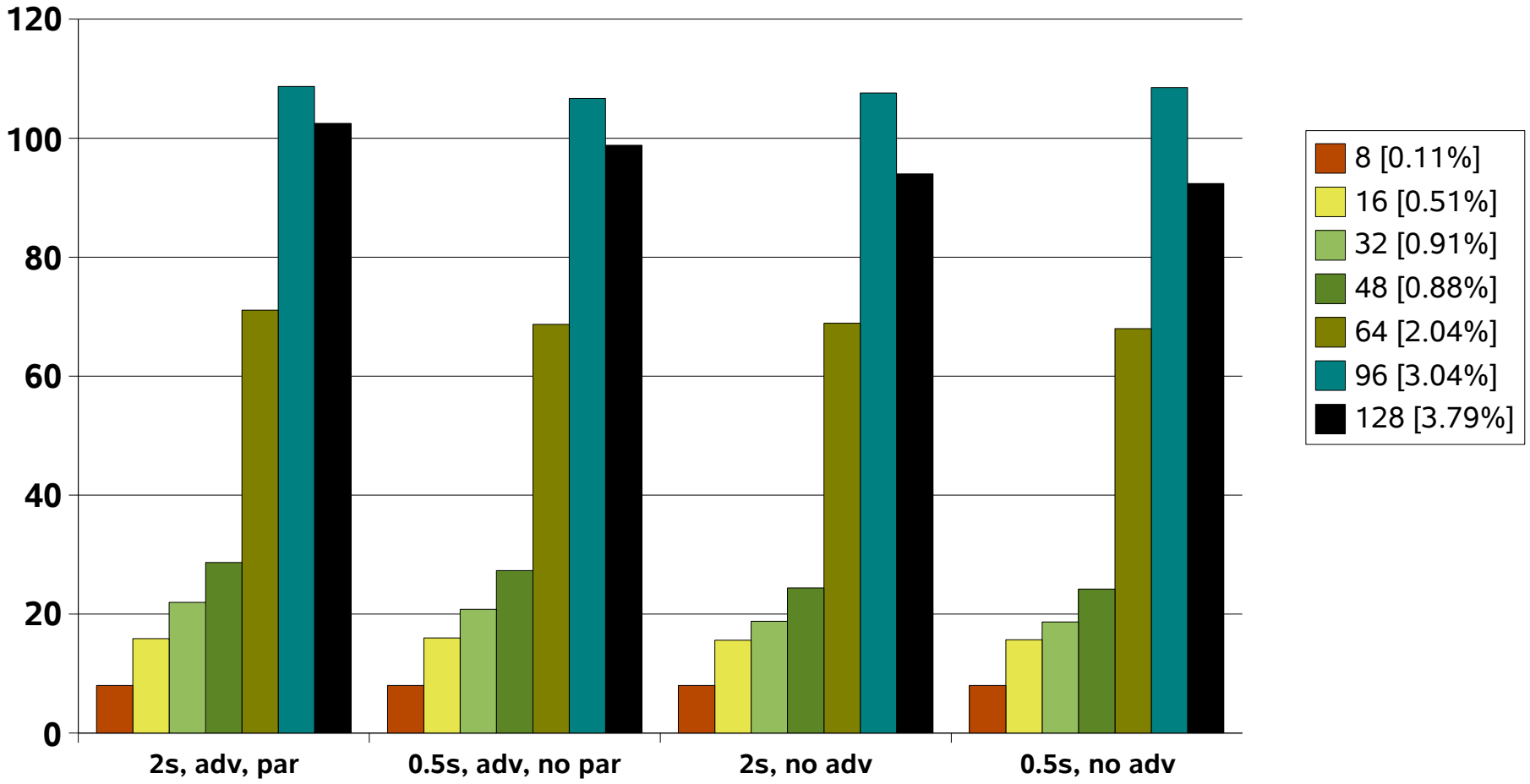
For the model The Elbe River by Coswig

Speedup achieved in the advection loop of the subroutine explicit

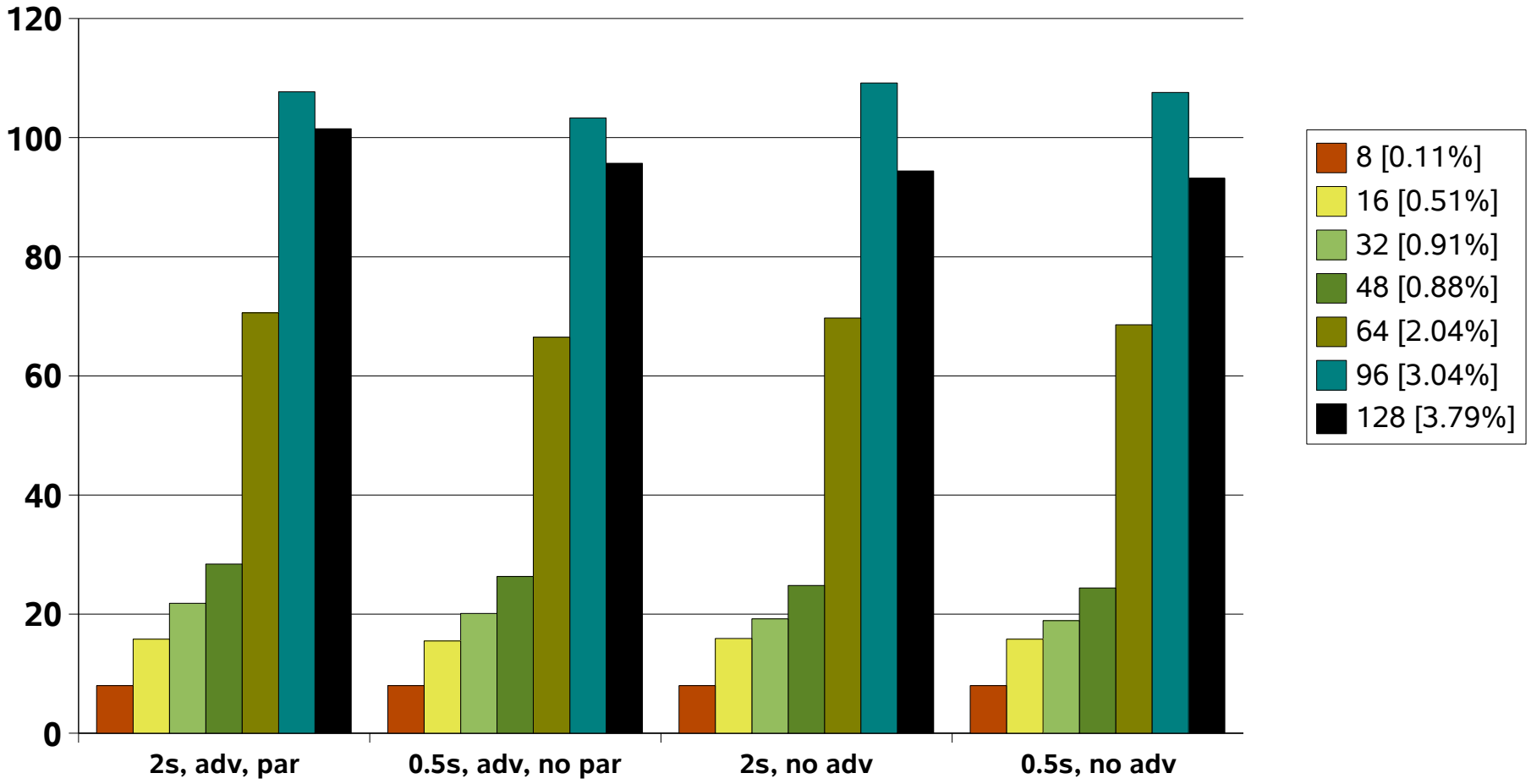
Coswig, advection speedup relative to 1p. (middle water, 2D, ne=738485)



Coswig, advection speedup relative to 8p. (middle water, 3D hyd, n3e=8006838)



Coswig, advection speedup relative to 8p. (middle water, 3D, non-hyd, n3e=8006838)



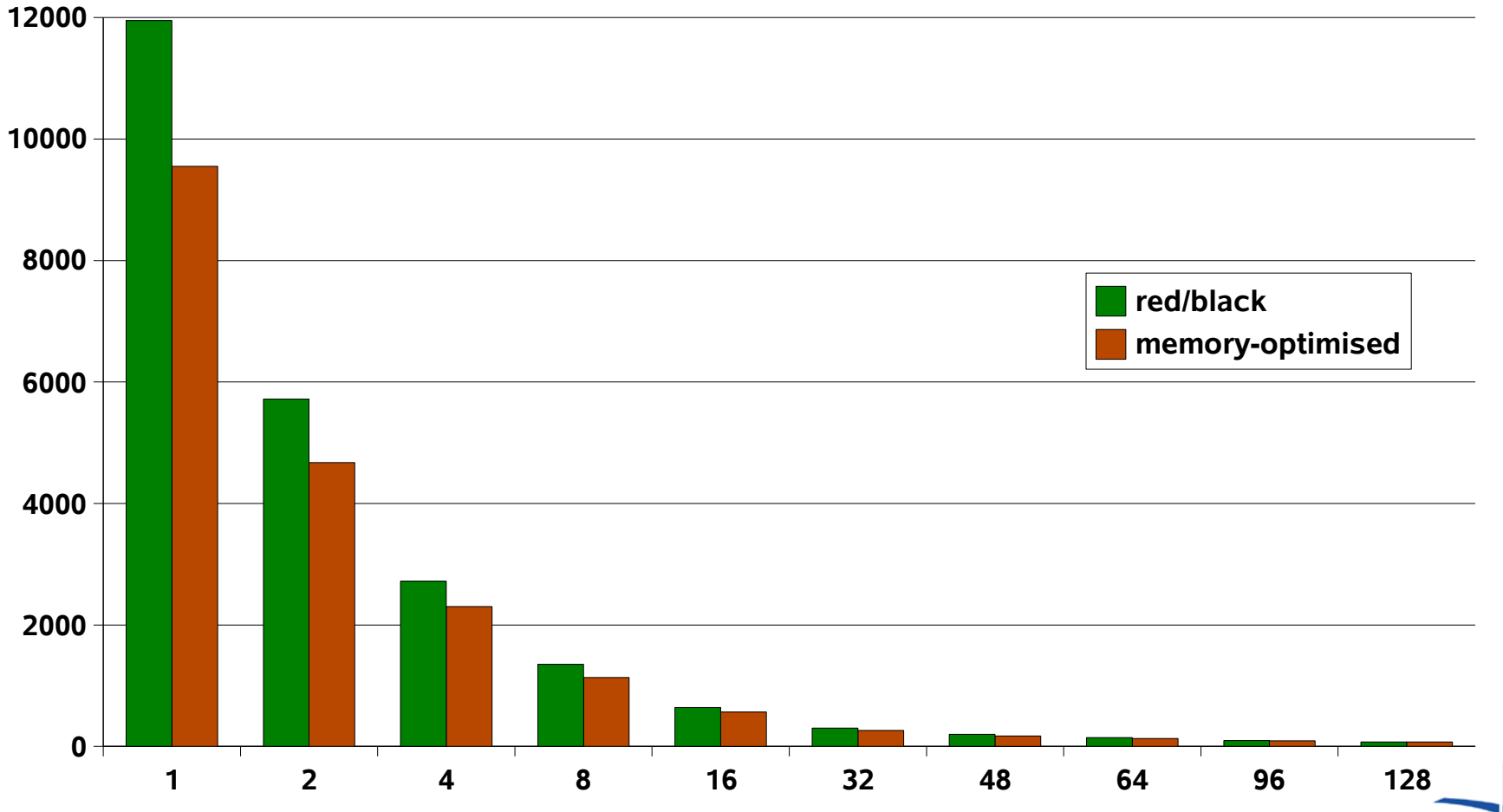
Examples

various transparencies for additional examples

Coswig mesh

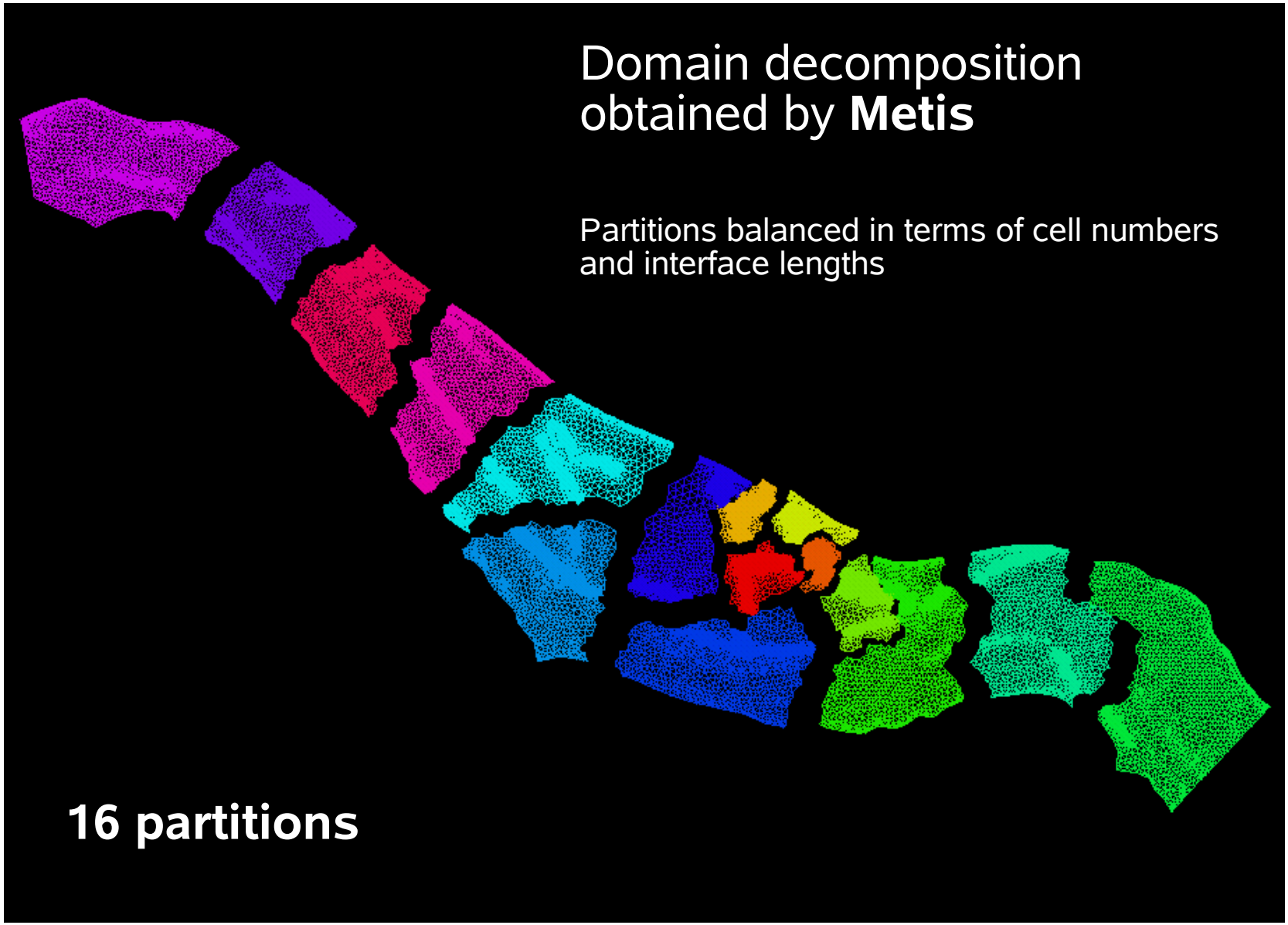
```
CoswigQH-mem     nk =        22   nv = 743882
ns =   1482366   ne =    738485   nbc = 102
n3s = 16149014   n3e = 8006838   ncs = 0

  0.645       < A  <    10.083
  0.441       < dx <    3.296
  0.519       < dy <    4.275

       1112    dx < 1m
    1468496    dx < 10m
    1468496    dx < 100m
```

Mesh sorting influence



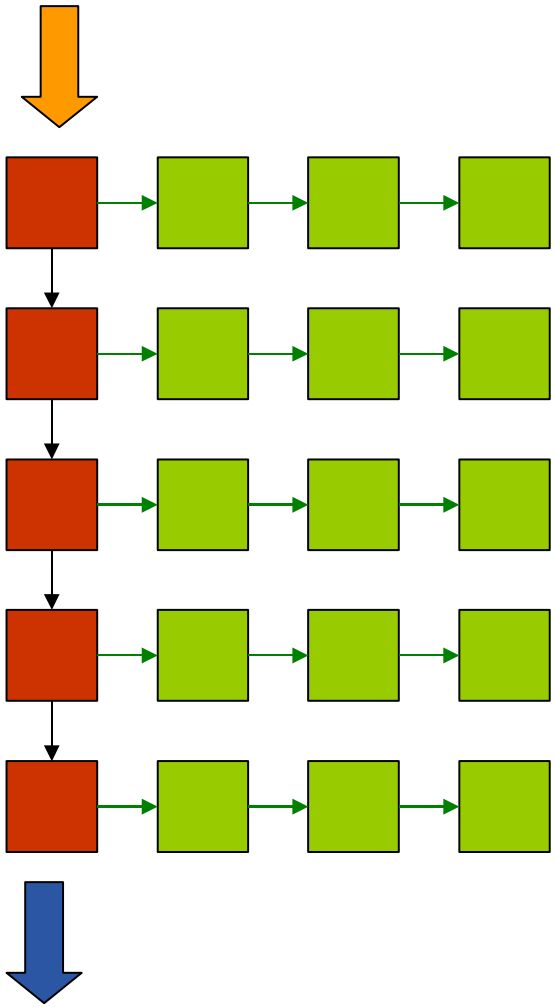
Domain decomposition obtained by **Metis**

Partitions balanced in terms of cell numbers and interface lengths



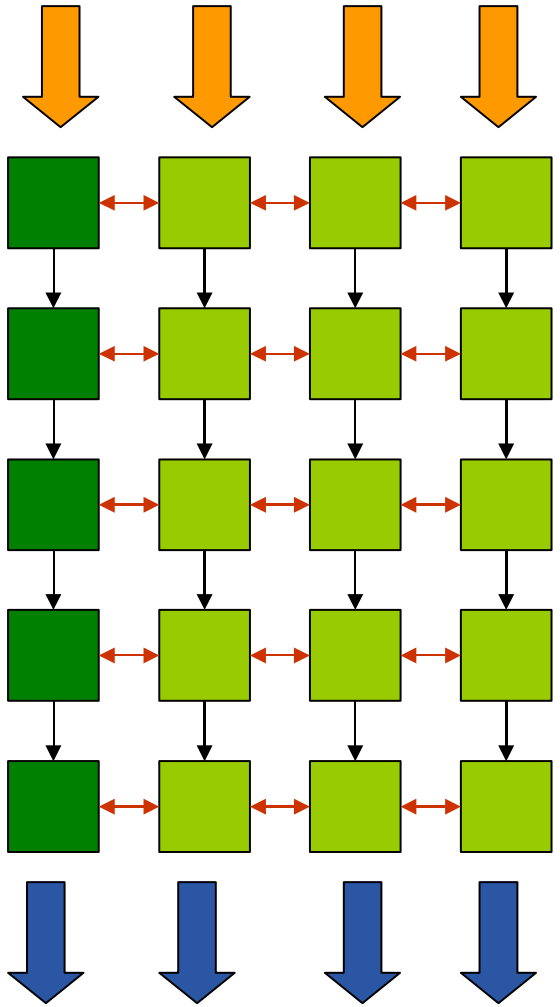
16 partitions

```
export OMP_NUM_THREADS=4
./prog.exe
```



OpenMP-run

```
mpirun -np 4 ./prog.exe
```

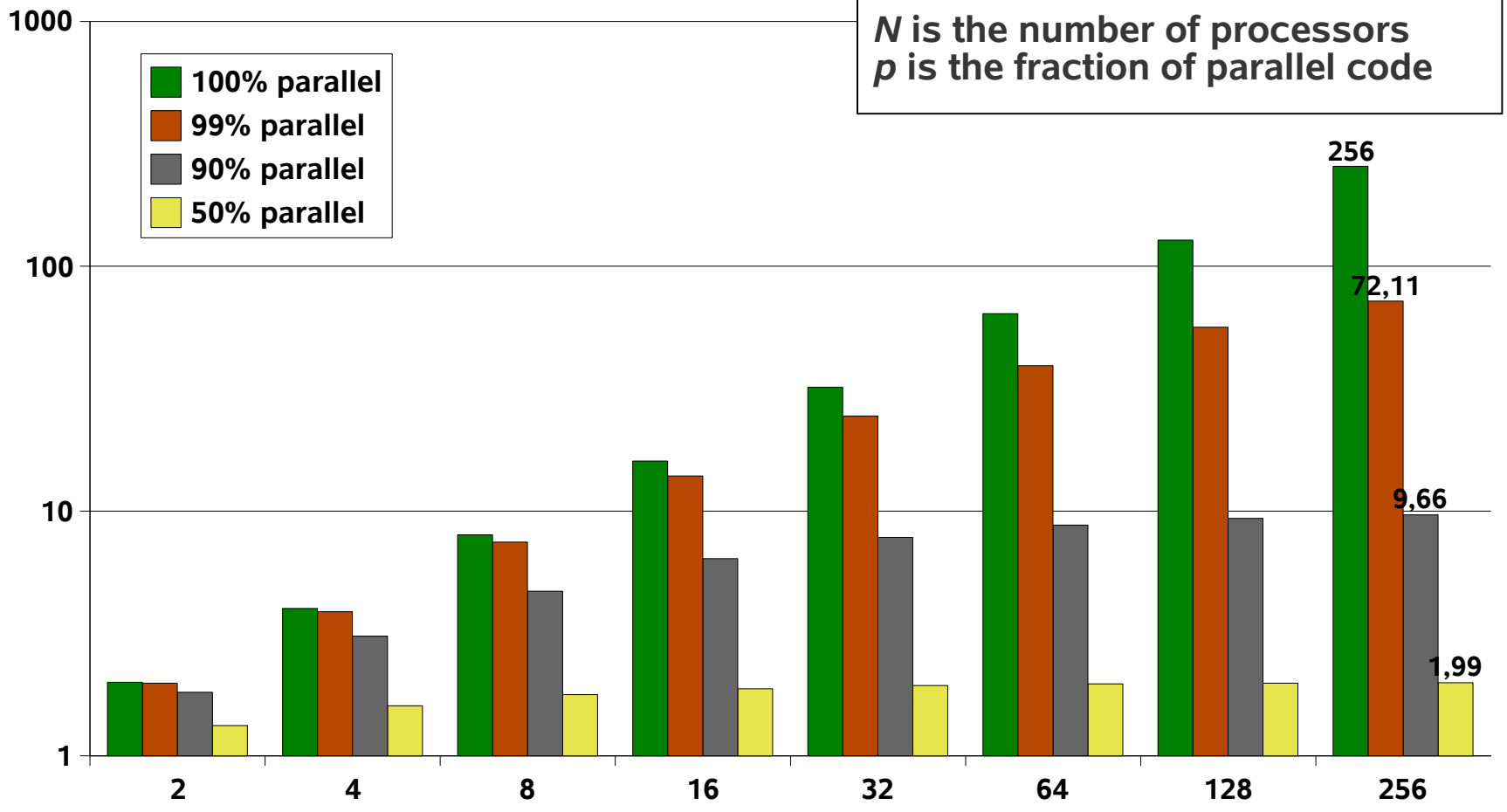


MPI-run

Amdahl's Law

Speedup (*N*) = 1 / (1 – *p* + (*p* / *N*))

***N* is the number of processors**
***p* is the fraction of parallel code**



Iterative UnTRIM parts

horizontal and vertical velocity:
iterative treatment of the explicit horizontal viscosity terms: **HVIS, DELT**

preliminary water level:
Np linear equations, preconditioned conjugate gradient: **EPSI**

hydrodynamic pressure:
Np*Nz linear equations, preconditioned conjugate gradient: **QEPSI**

species transport:
an iterative algorithm for the transport treatment
DTMIN, DELT

In all these algorithm parts a special care must be taken for the communication inside the iterations

machone.karlsruhe.baw.de

- SGI Altix 350
- **four** 1400 MHz Itanium-2 processors, 3MB cache
- 4GB shared memory
- 64-bit Linux
- Intel and Gnu compilers
- OpenMP and MPI

Project Diary