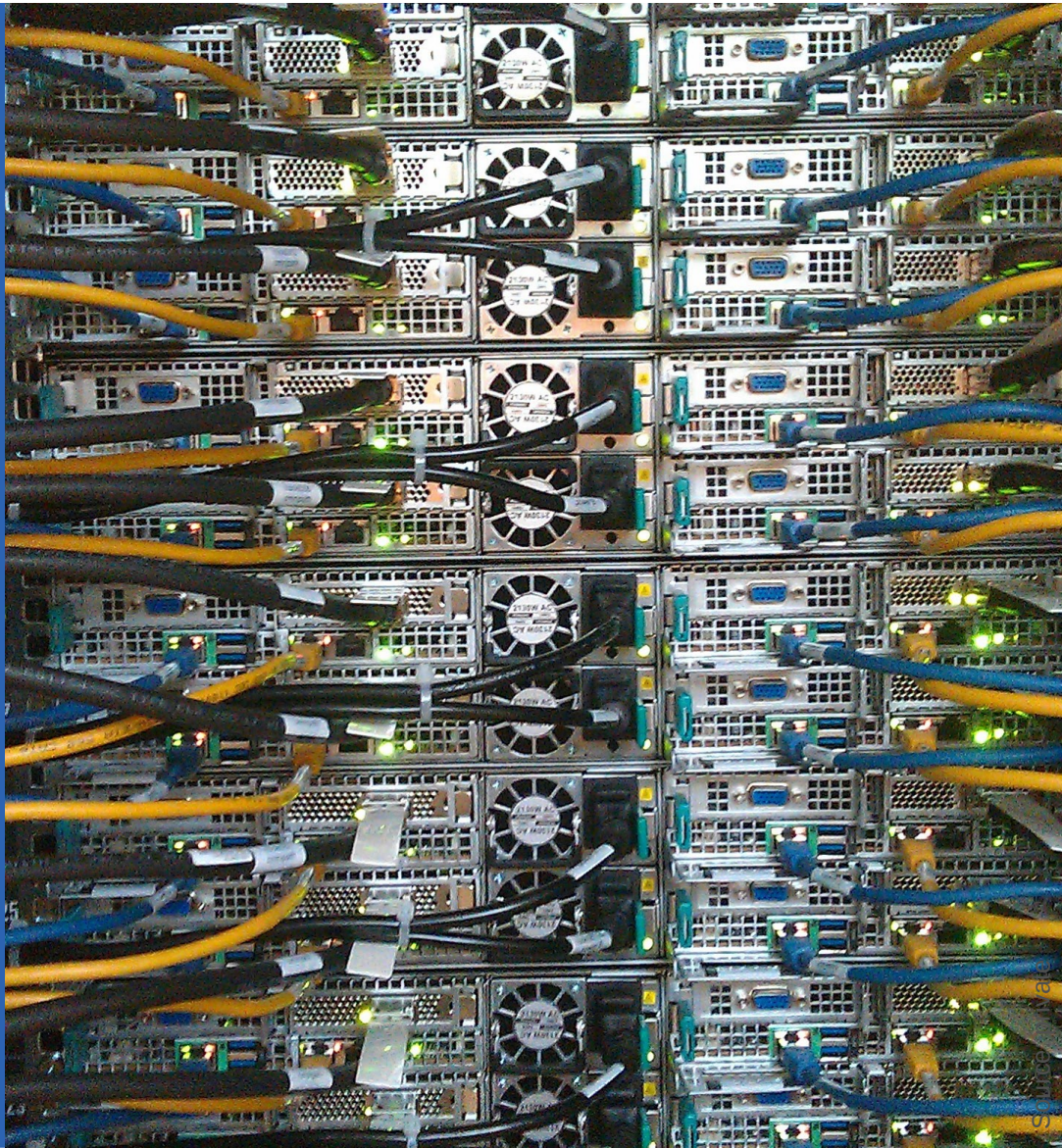
BAW
Federal Waterways Engineering and Research Institute

Jacek A. Jankowski

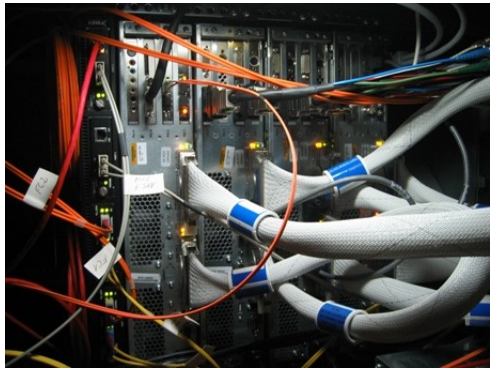
Ten Years After: MPI-UnTRIM’s past and future

15th UnTRIM Workshop

Trento, 28-30 May 2018



Content



1. A melancholic introduction
2. The domain decomposition method [2005]
3. Swapping point-to-point communication [2006]
4. Assembling communication – diffusion operator [2008]
5. Parallel streamline tracking, all-to-all communication [2007]
6. *If there‘s enough time*: Parallel I/O [2010] and HDF5, netCDF in parallel context
7. A philosophic outlook

Jacek A. Jankowski

Ten Years After: MPI-UnTRIM’s past and future

15th UnTRIM Workshop

Trento, 28-30 May 2018

Source: openclipart.org

Antoine de Saint-Exupéry knew it better

A goal
without a plan
is just a wish.

Antoine de Saint Exupéry by the wreck of his plane due from Paris to Saigon. Libyan Desert, 1935.

So
ur
ce
:
Wi
ki
pe
di
a

Based on the past activities presented already as…

UnTRIM parallelisation with message passing, 2nd workshop, Sirmione, 23-25 May **2005**

UnTRIM parallelisation with message passing, 3rd workshop, Trento, 15-17 May **2006**

Parallel streamline tracking for UnTRIM, 4th workshop, Trento, 7-9 May **2007**

Verification and validation of the parallel UnTRIM, 5th workshop, Trento, 19-21 May **2008**

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
- 2005

machone.karlsruhe.baw.de, a SGI Altix 350

obelix.karlsruhe.baw.de

- 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** [I mean in 2005…]

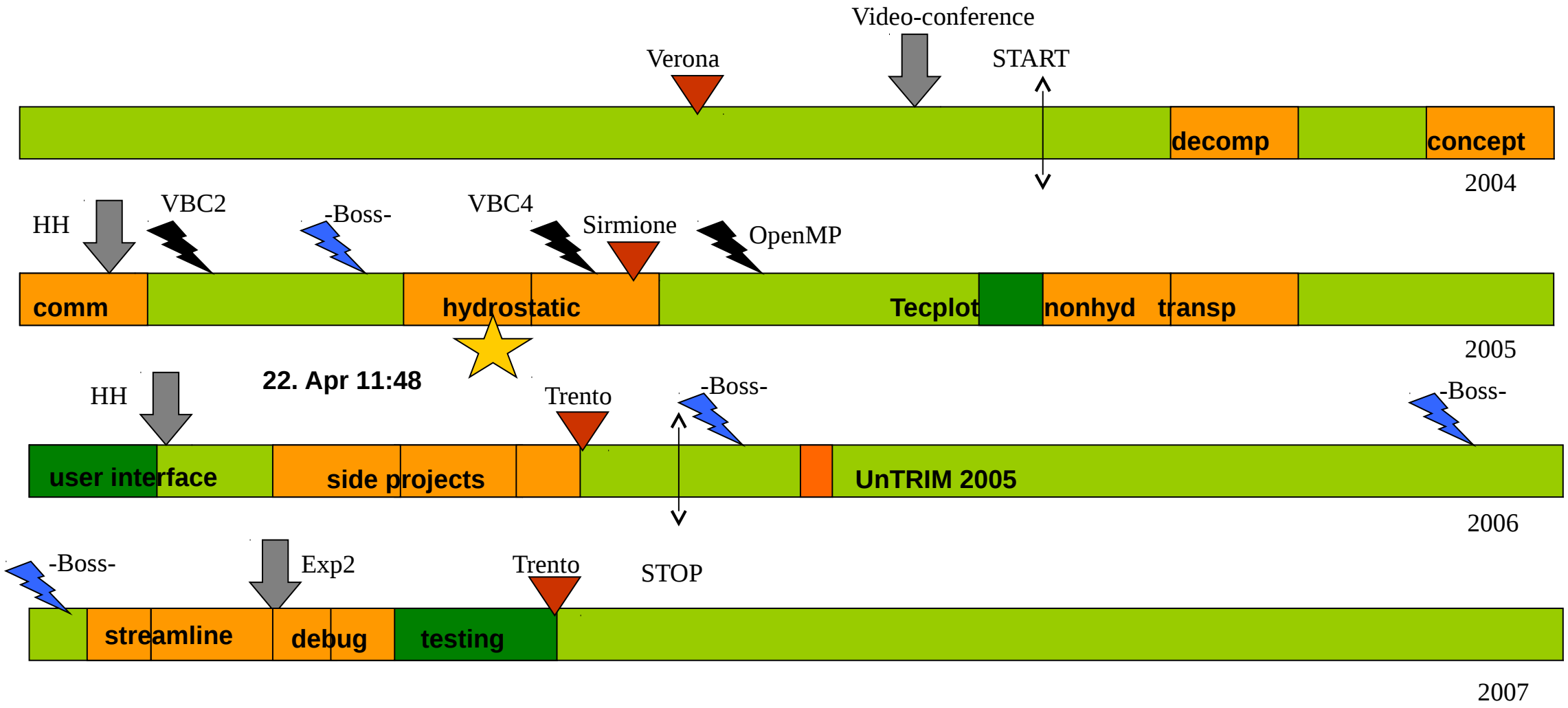
obelix.karlsruhe.baw.de, a SGI Altix 3600

„A talented Fortran programmer“ [Günther Lang]

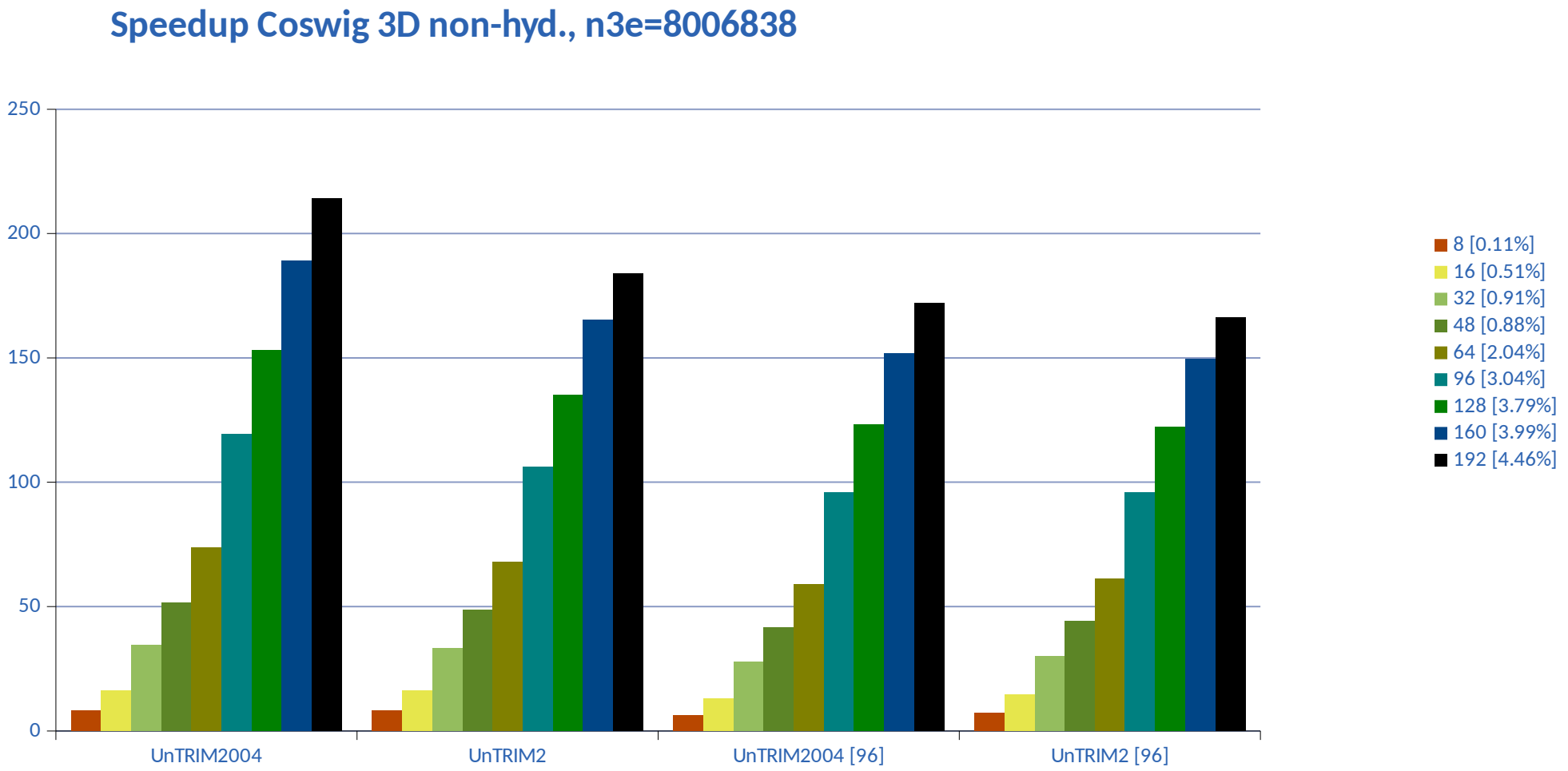
- 2004-2010 continuously in one organisatory unit (fluvial engng.)
- Could work approximately 1/3 of the formal work time programming in parallel UnTRIM
- In theory it was a co-operation between BAW coastal and inland waterways departments

2005

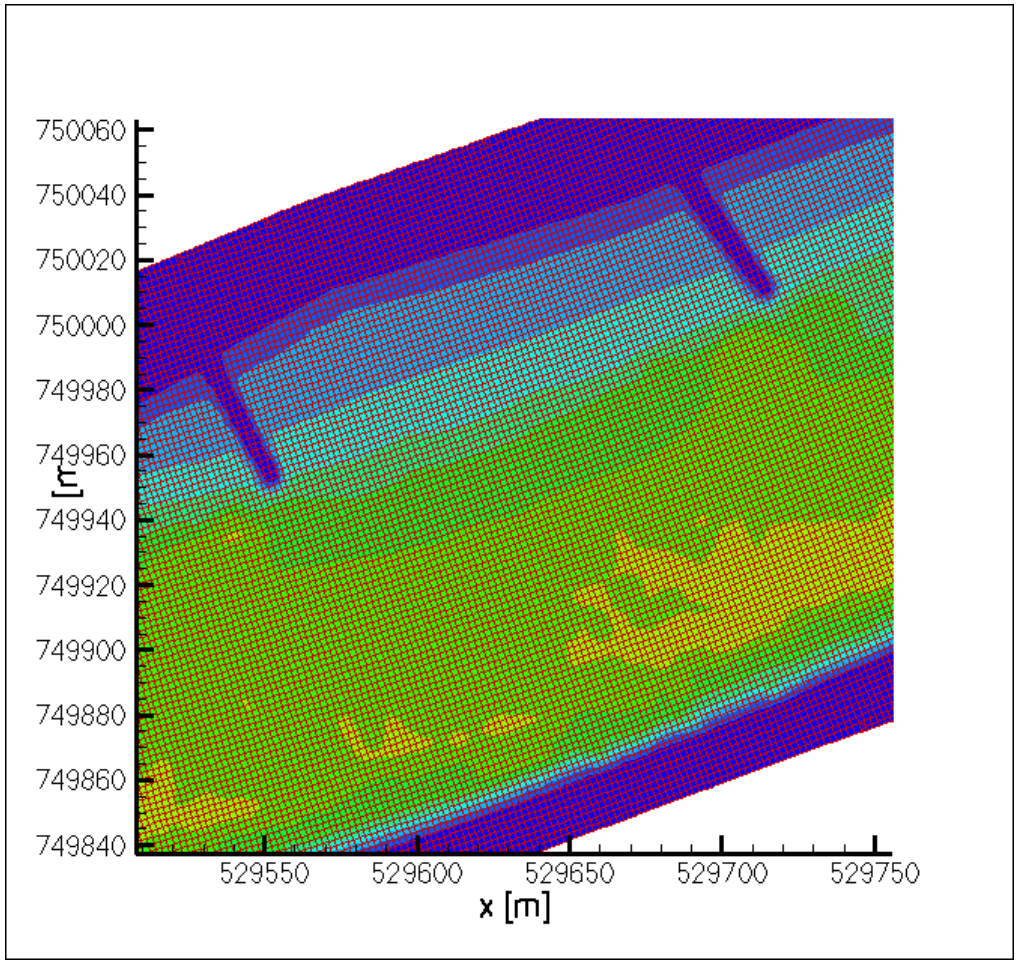
Project Diary from the 2007 presentation

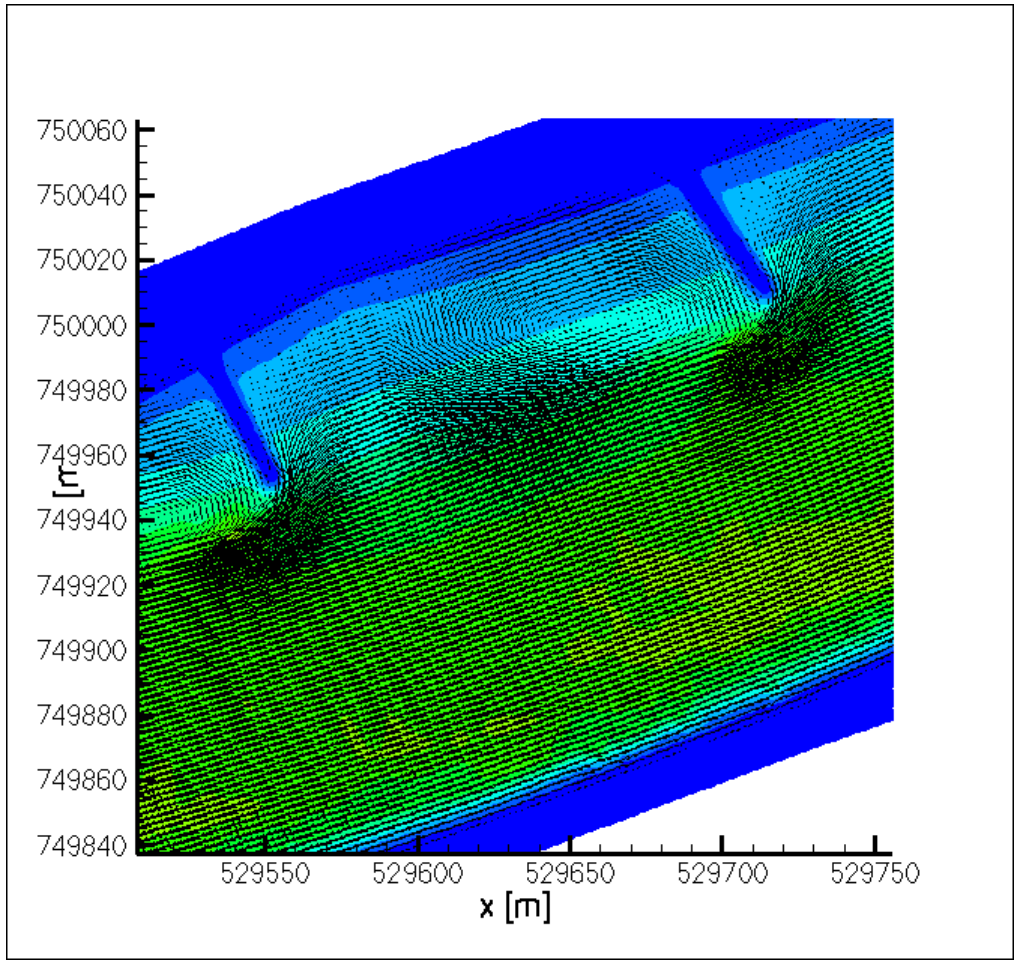


Speedup UnTRIM [2004] contra UnTRIM [2008]



High resolution models





The Elbe River by Coswig [2007] – 14km river stretch – a mesh of very regular quadrangles with dx=2m, dz=0.5m, ne=738485, nk=22, n3e=8006838

Applications & Developments 2007-2011

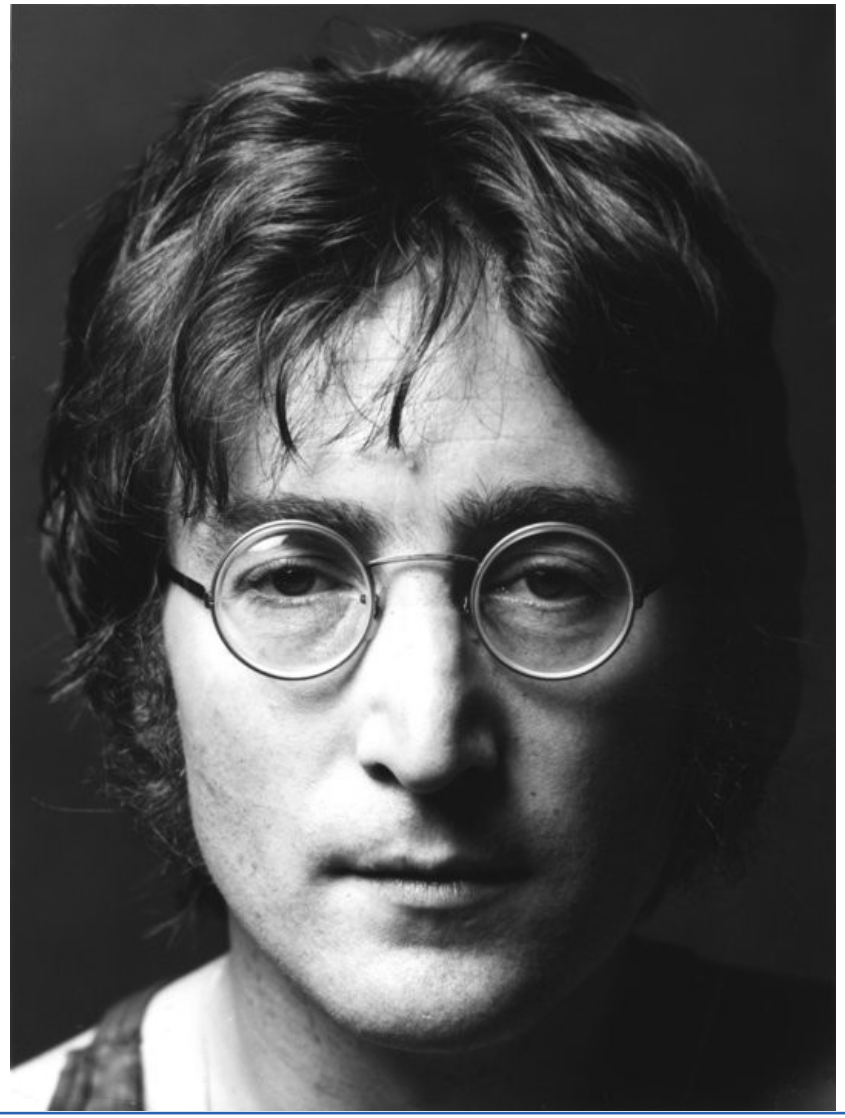
- The parallelised kernel with User Library specifically adapted for river engineering started to be applied since ca. 2007

- Quite a number of practical applications, just to name:
  - The Elbe River – stretches by Coswig, Lenzen, Hitzacker, Geesthacht…
  - The Danube River – Mühlhamer Loop, 80km-stretch, the Isar confluence…
  - The Saale River

- Additional developments in this period concerned
  - the scope of physical processes included (roughness, turbulence…)
  - pre- and postprocessing, mesh treatment, parallel I/O (not applied)

John Lennon knew it well

Life is what happens
when you are busy
making other plans.



[Source: Facebook]

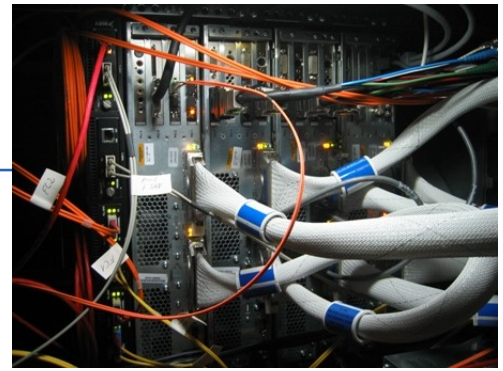
Applications & Developments 2011-today

- Developments:
  - working with GPUs, MICs, vectorisation, parallel I/O in UnTRIM
  - „(Un)TRIM-like“ 2D-schemes with GPUs 2010-11
  - MPI-parallelised UnTRIM2 (subgrids) spring 2013

- 2014: Decision to limit the methodological scope by inland waterways to Telemac only
  - only a study for subgrids feasibility assessment [2016]

- Initiatives to profit from the past developments in 2016 and 2018
  - UnTRIM3 (groundwater flows) to be MPI-parallelised soon…

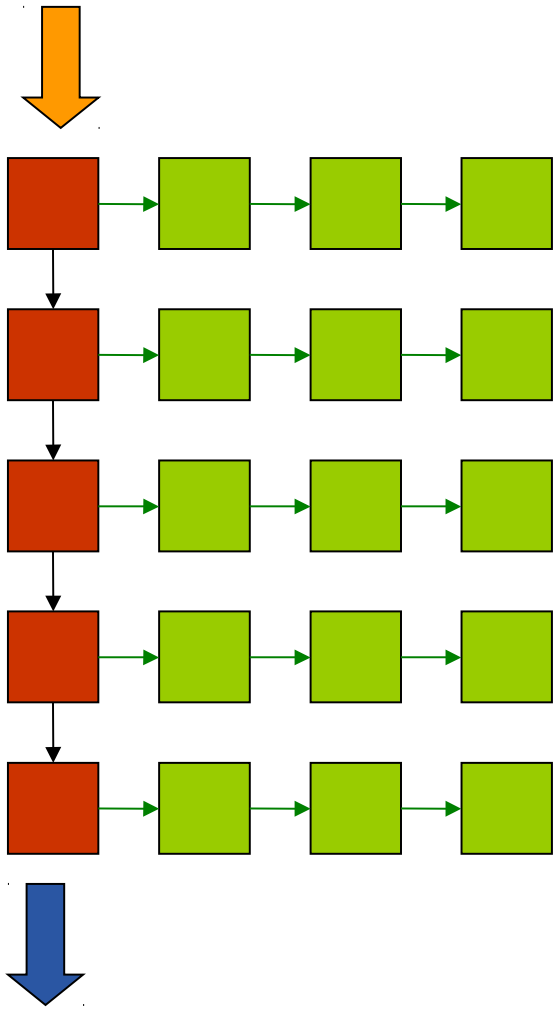
Hand made for a parallel machine



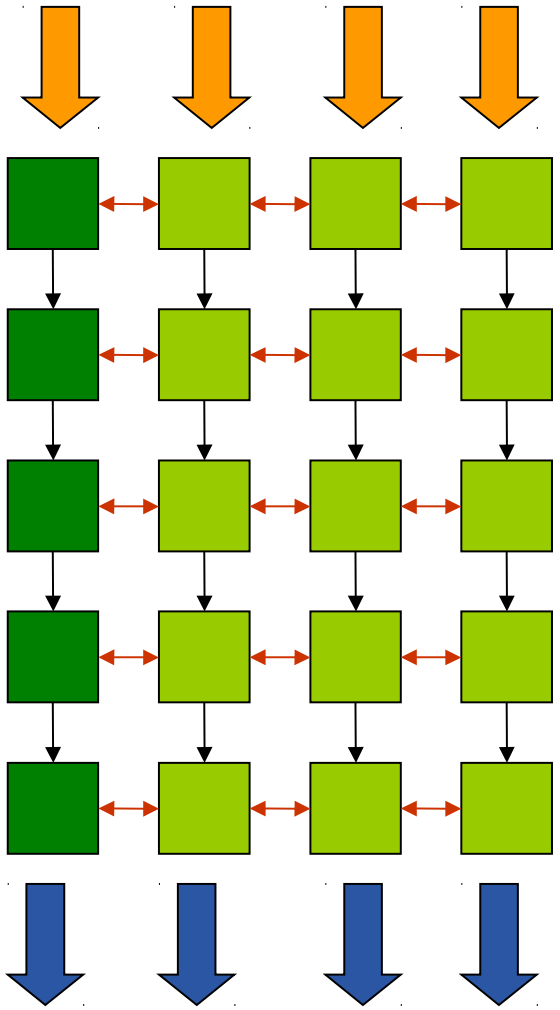
- **Domain decomposition method** – partitioning input data and merging output, synchronous or asynchronous communication
- **Message-passing** – optimising the communication for efficiency
- **Standard swapping communication** – for ~90% of the communication
- **Tailored „hand-made“ solutions for** –
  - explicit diffusion operator – assembling contributions
  - Lagrangian advection scheme – streamline backtracking over partitions, global communication
- **Verification, validation, optimisation** – evidence of serial results conformity
- Some **node-level optimisations**

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

```
mpirun –np 4 ./prog.exe
```



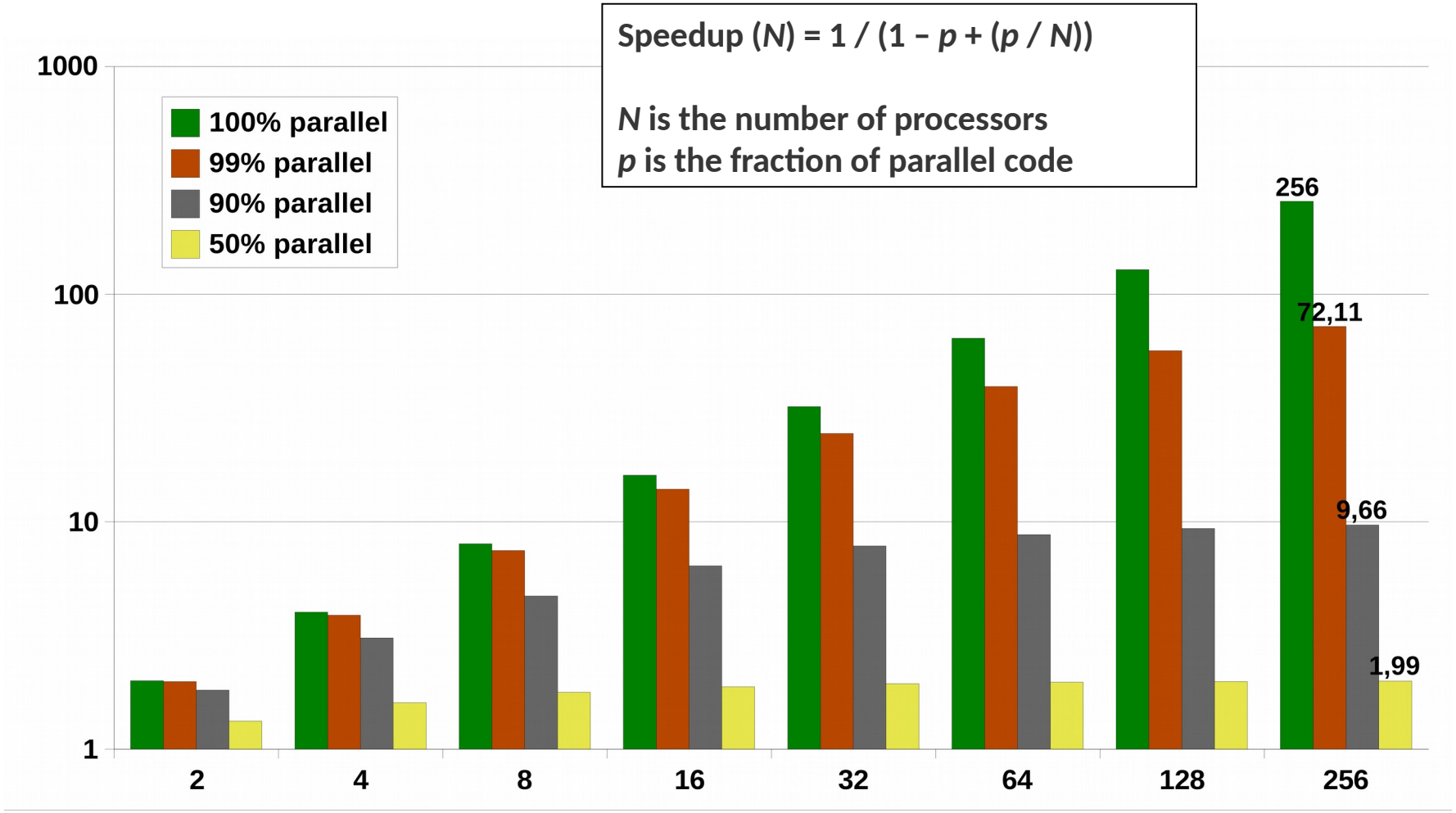
OpenMP-run



MPI-run

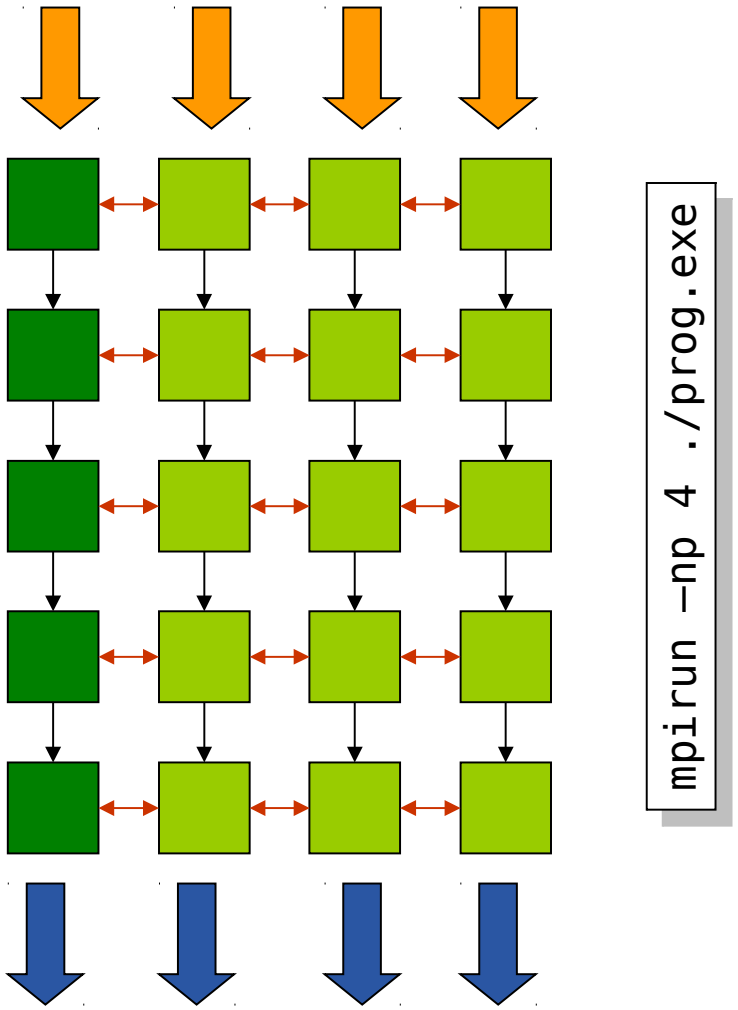
The principle parallel execution modes

Amdahl´s Law


Speedup (N) = 1 / (1 – p + (p / N))
N is the number of processors
p is the fraction of parallel code
100% parallel
99% parallel
90% parallel
50% parallel
1000
100
10
1
2
4
8
16
32
64
128
256
256
72,11
9,66
1,99

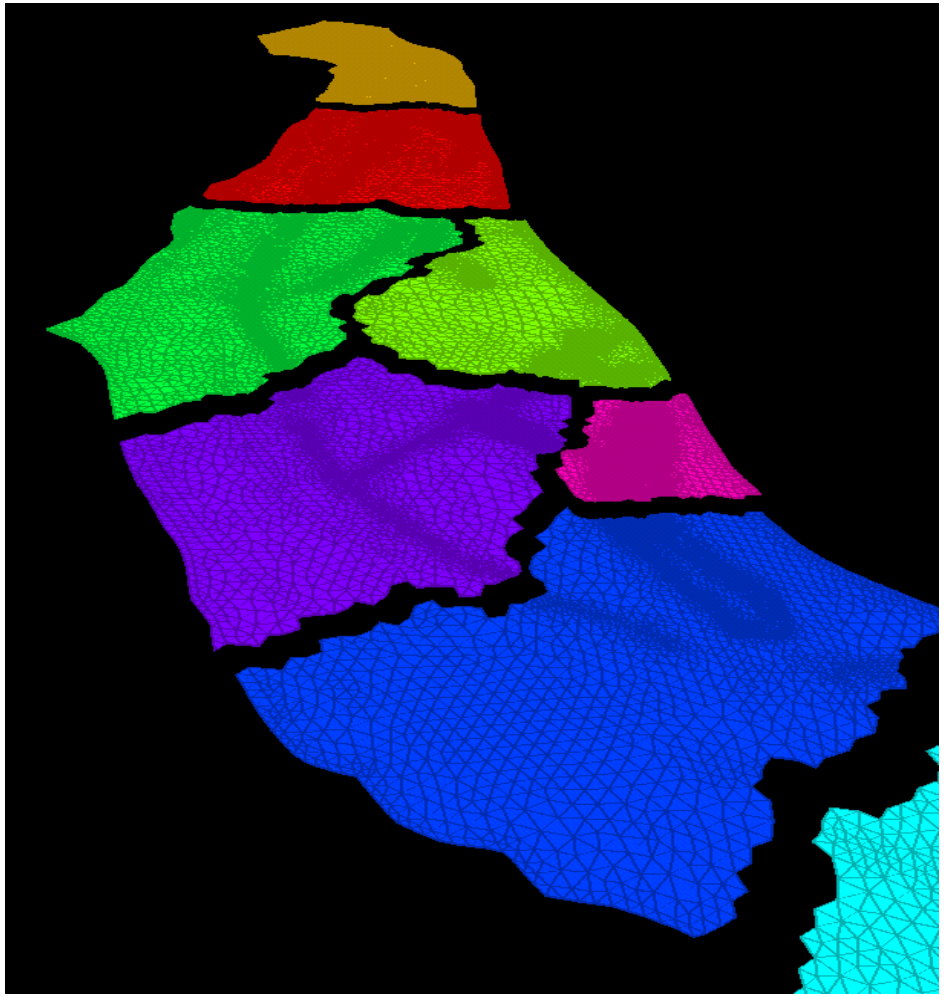
Message-passing parallelism

- 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



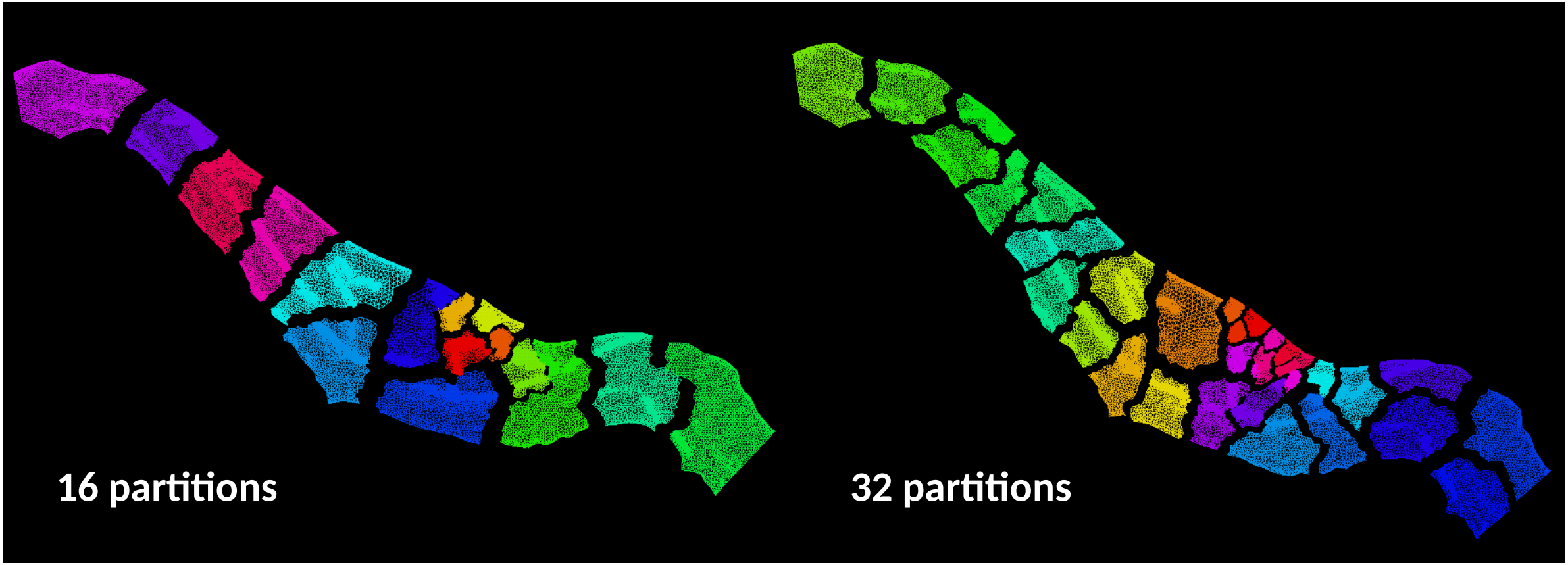
Domain decomposition method

- Parallel implementation with domain decomposition and overlapping mesh partitions (FDM, FVM, FEM)
- This leads to point-to-point swapping communication between neighbouring partitions
- Explicit diffusion operator requires assembling communication
- Semi-Lagrangian advection methods requires special treatment

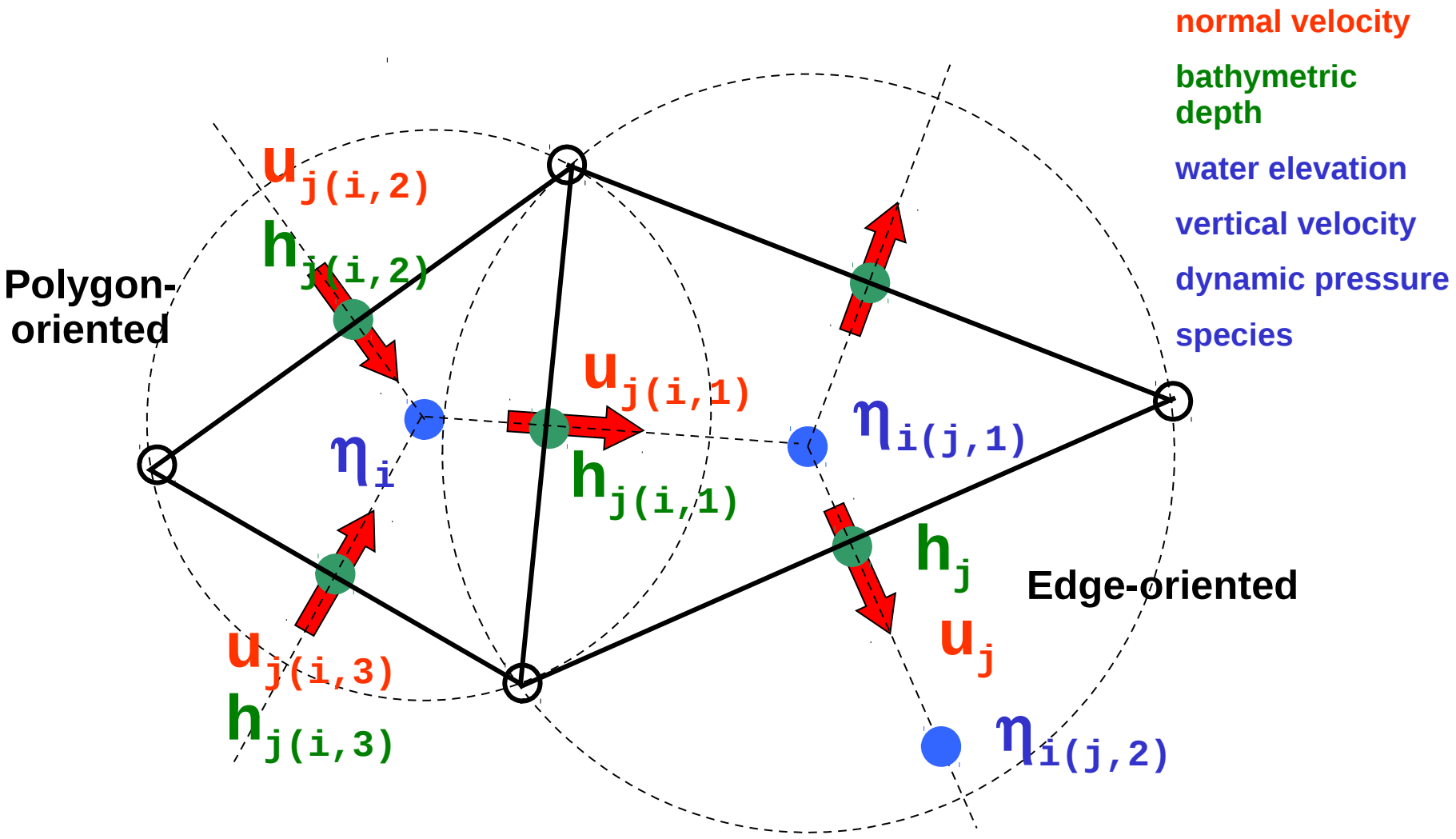


Balanced mesh partitioning (with Metis library)

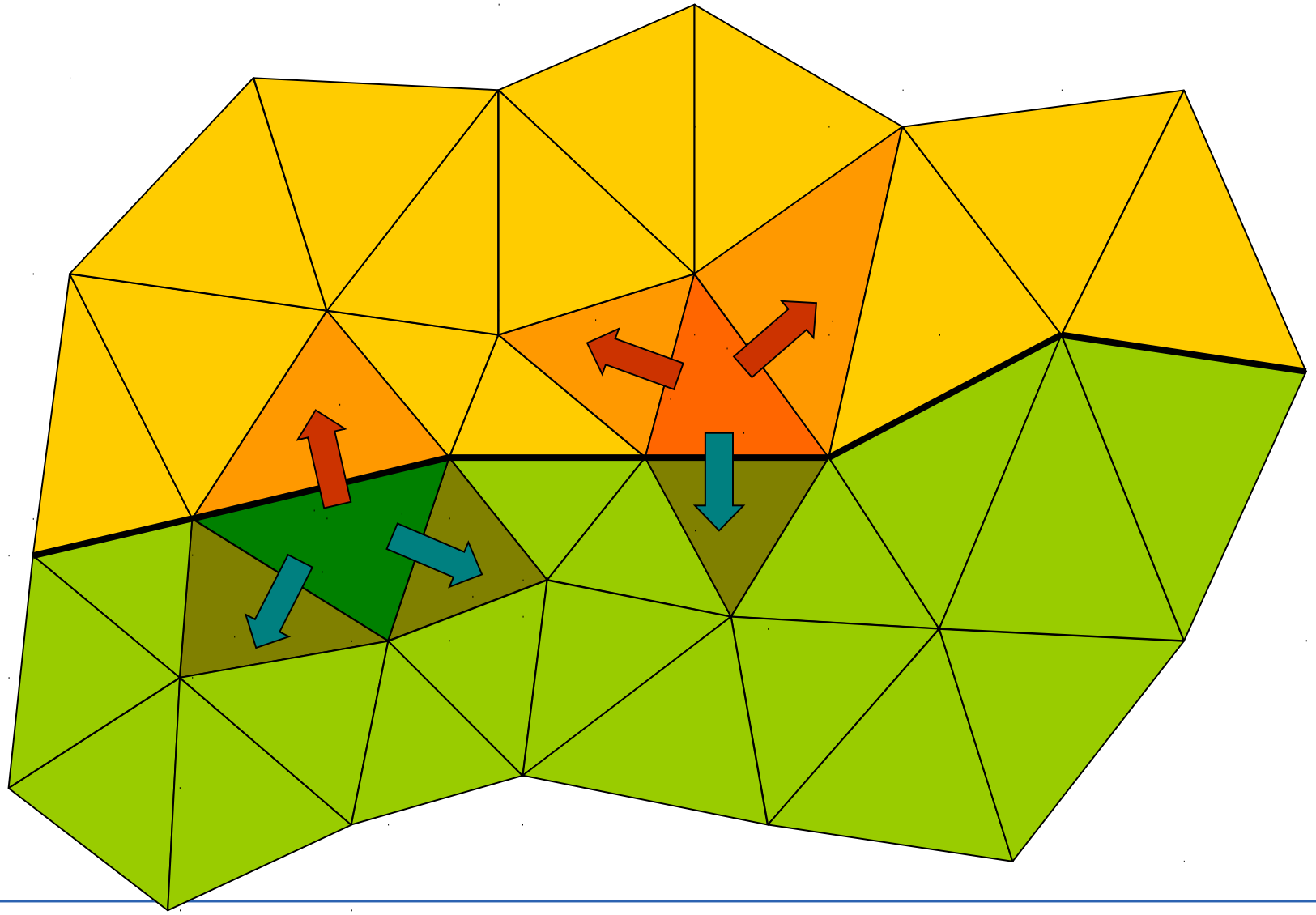
Partitions balanced in terms of mesh cell number and interface lengths



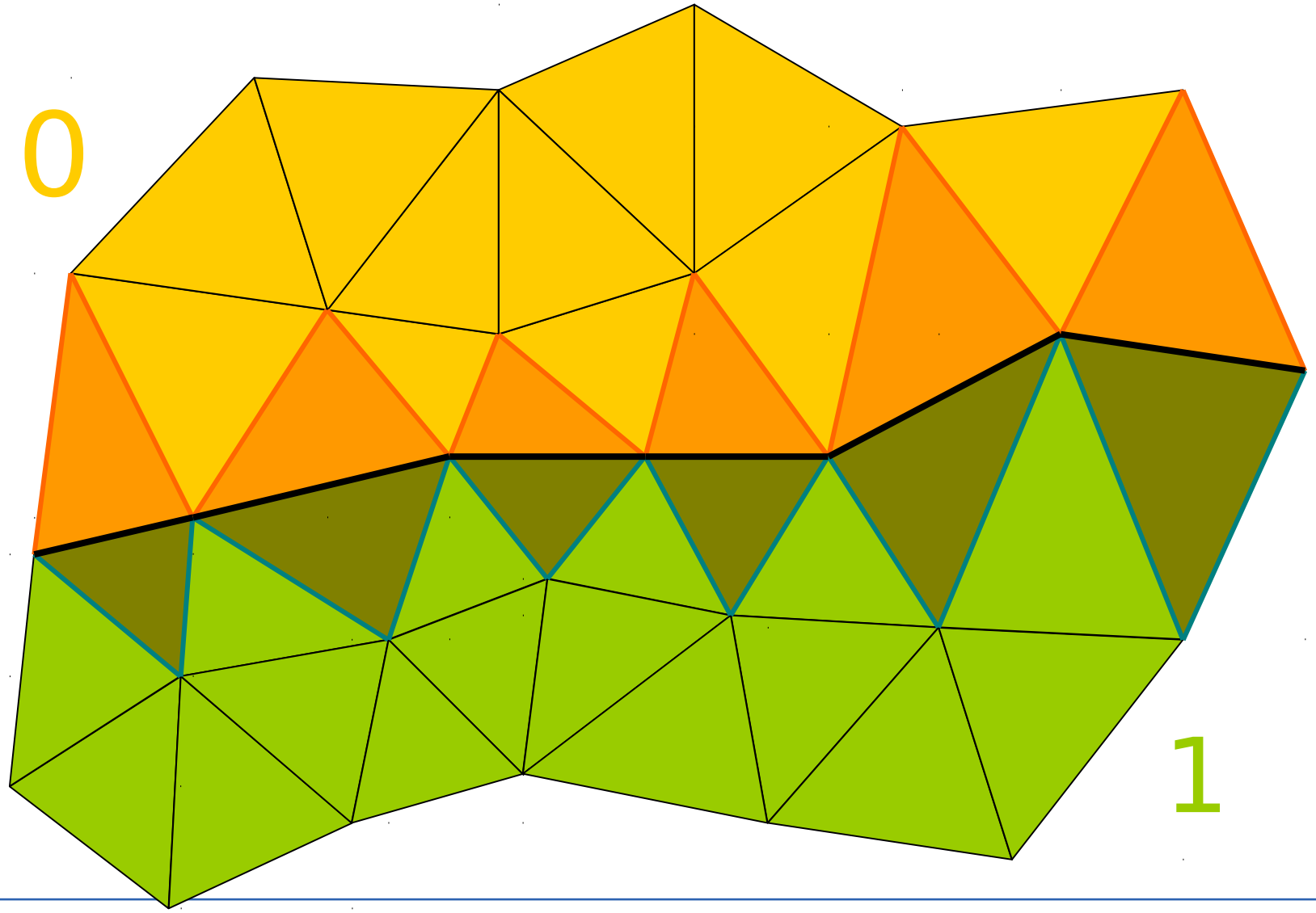
Horizontal structure – stencils



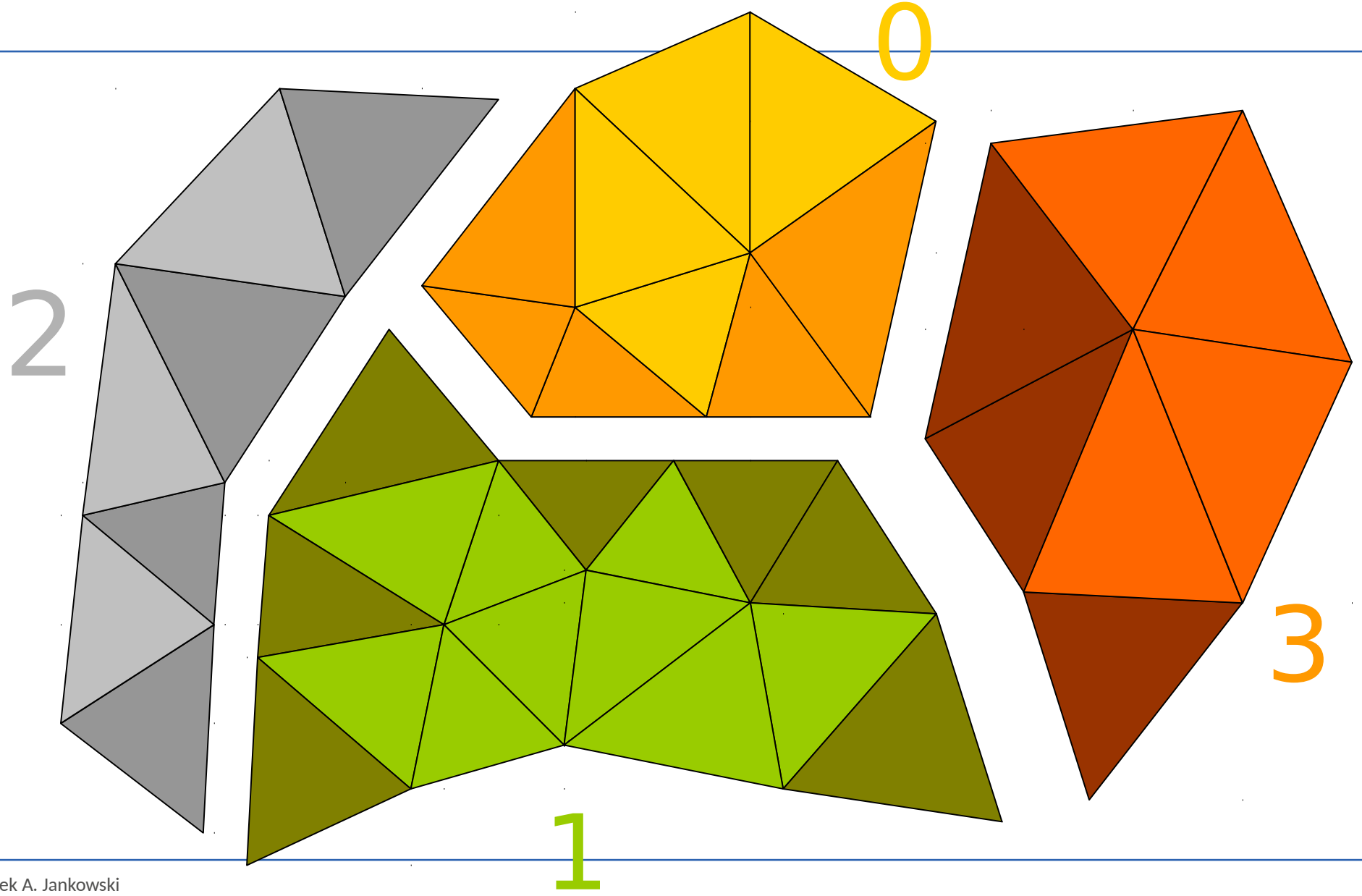
Dependency patches – stencils – define halos



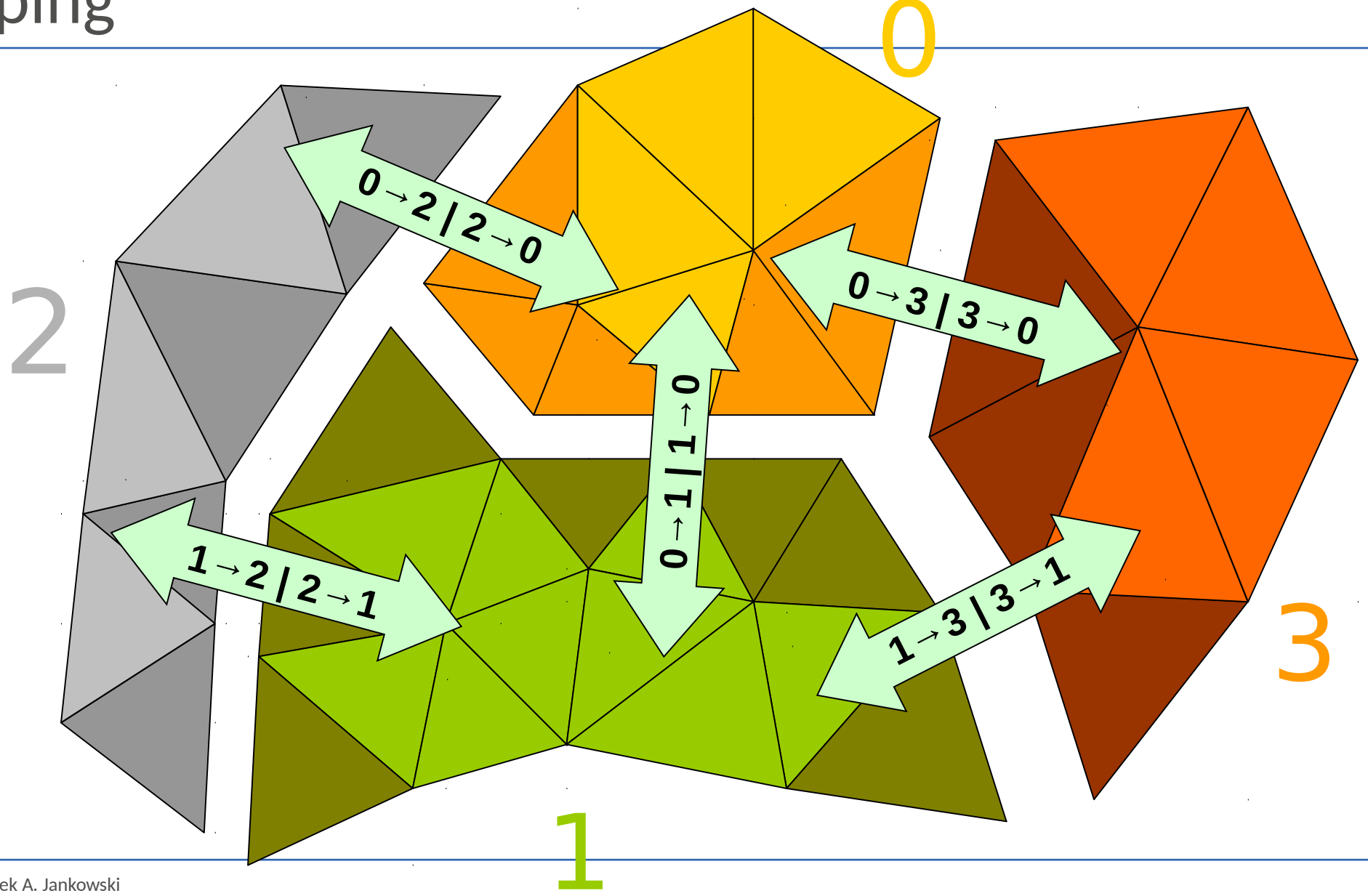
Halos: polygons and edges



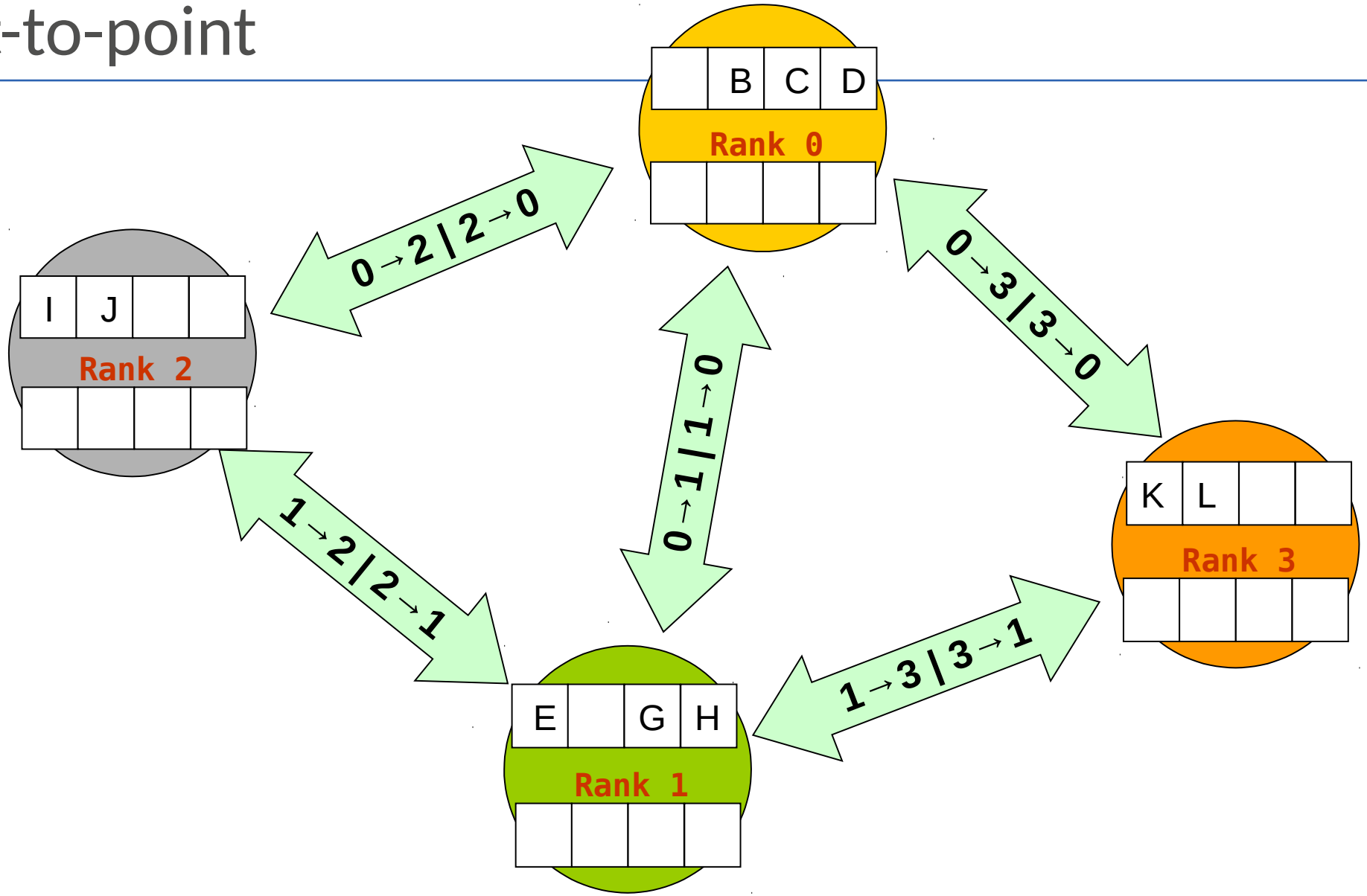
Halos



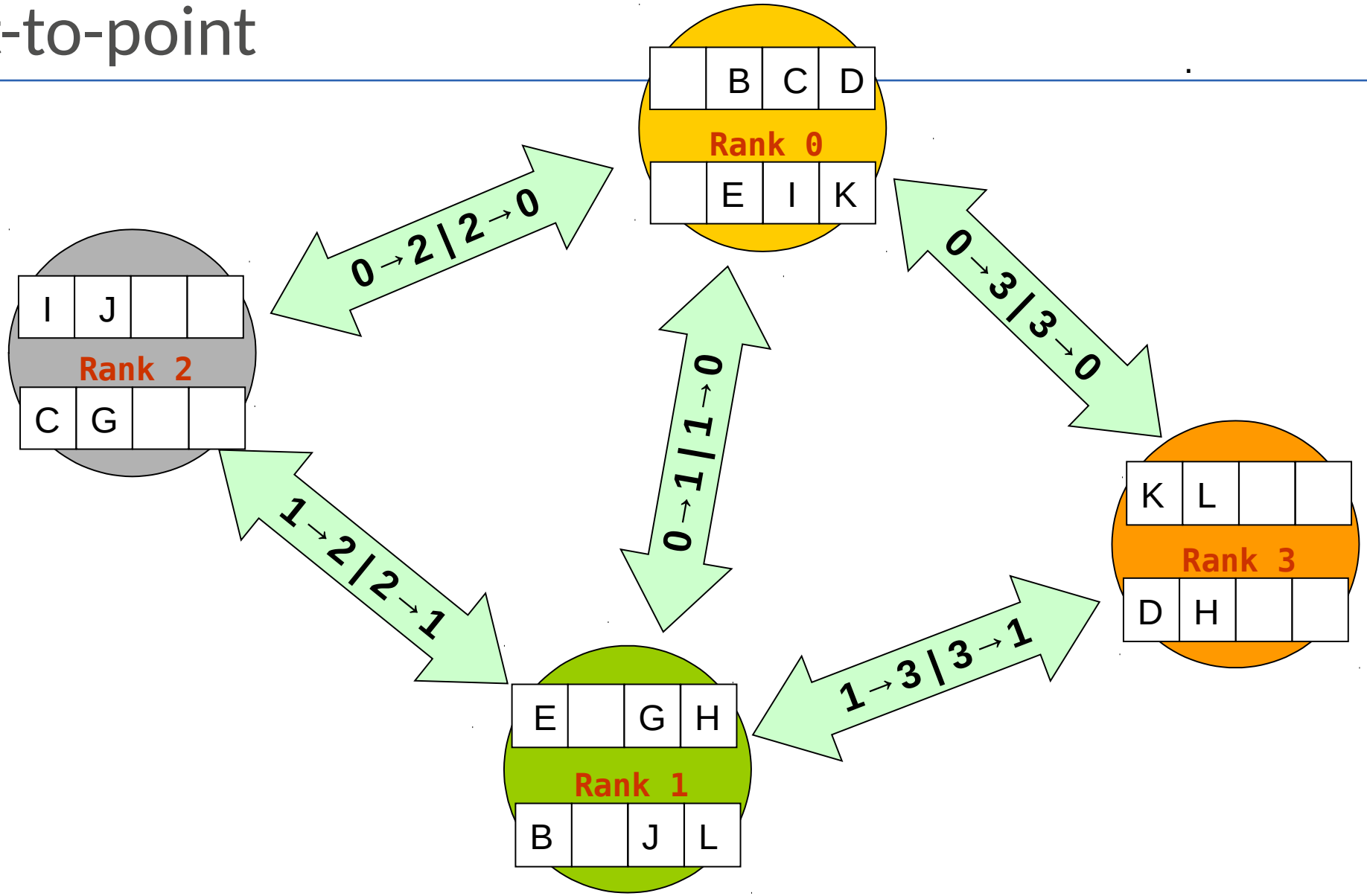
Swapping



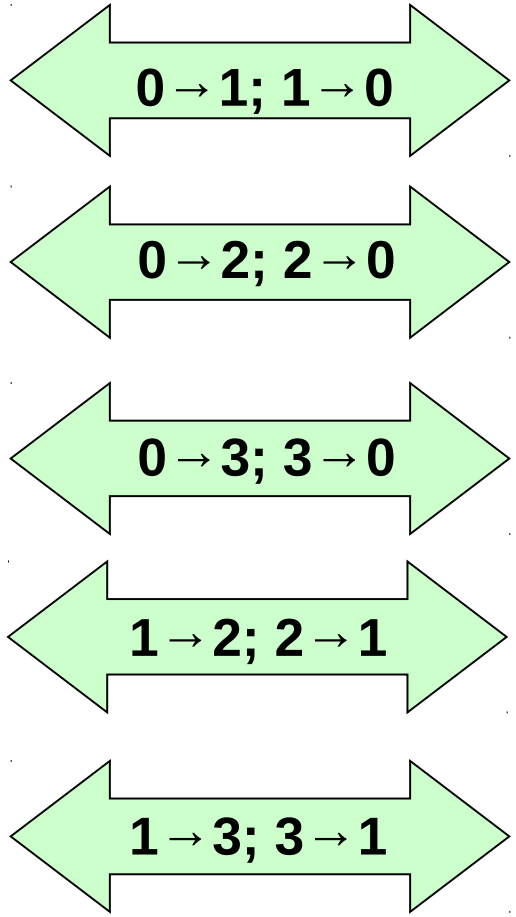
Point-to-point



Point-to-point

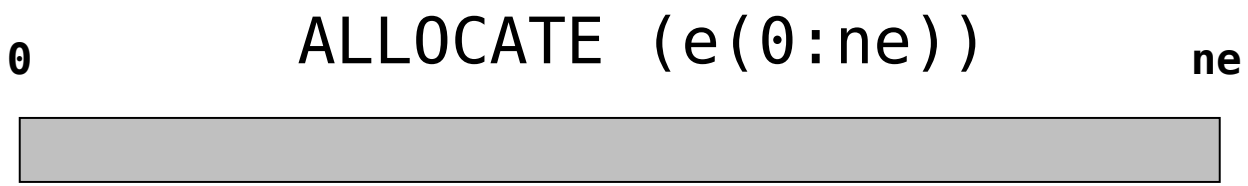


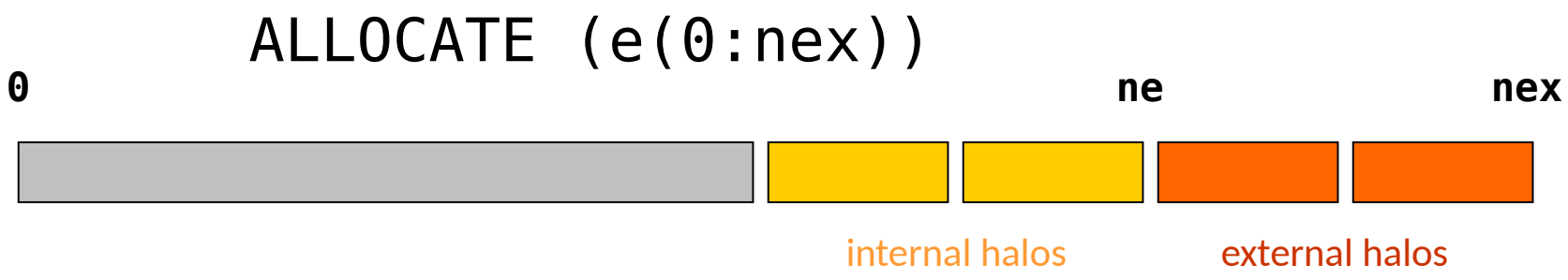
MPI_SendRecv



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

Extended fields: cells

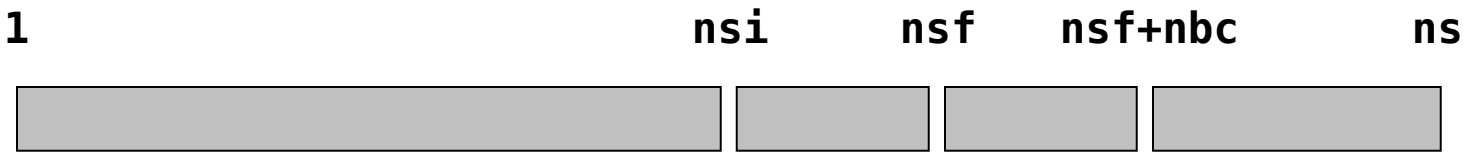




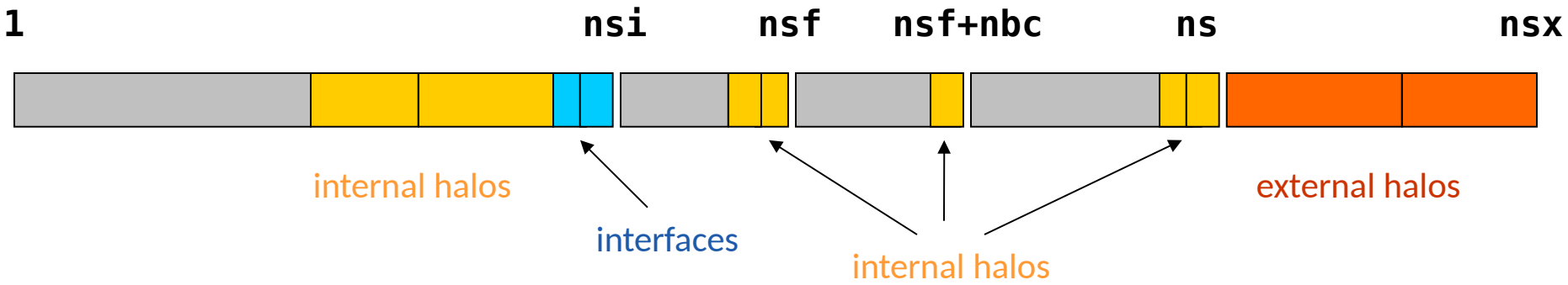
For 3D: `ALLOCATE(q(1:n3ex))`

Extended fields: edges

```
ALLOCATE (u(1:ns))
```

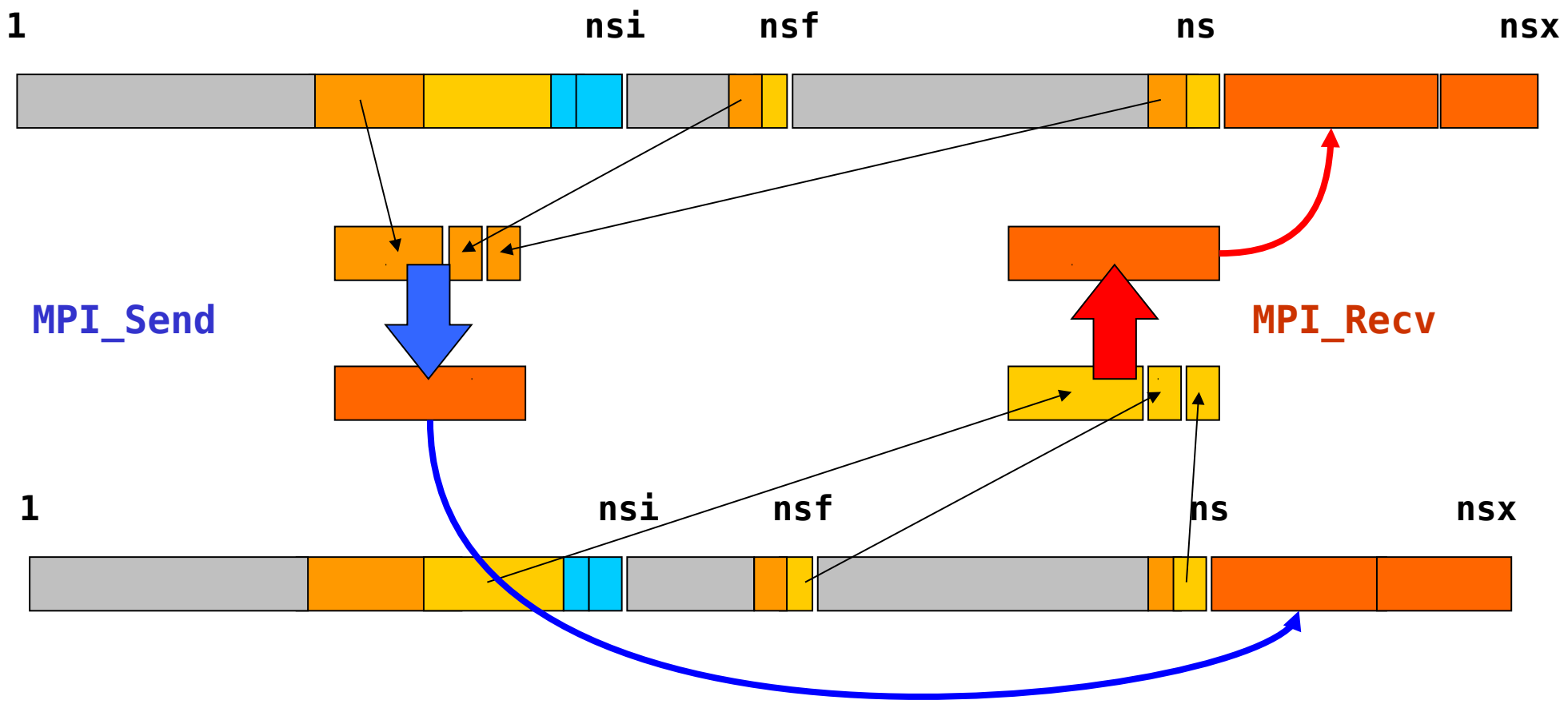



```
ALLOCATE (u(1:nsx))
```



For 3D: `ALLOCATE (u(1:n3sx))`

MPI_SendRecv with buffers

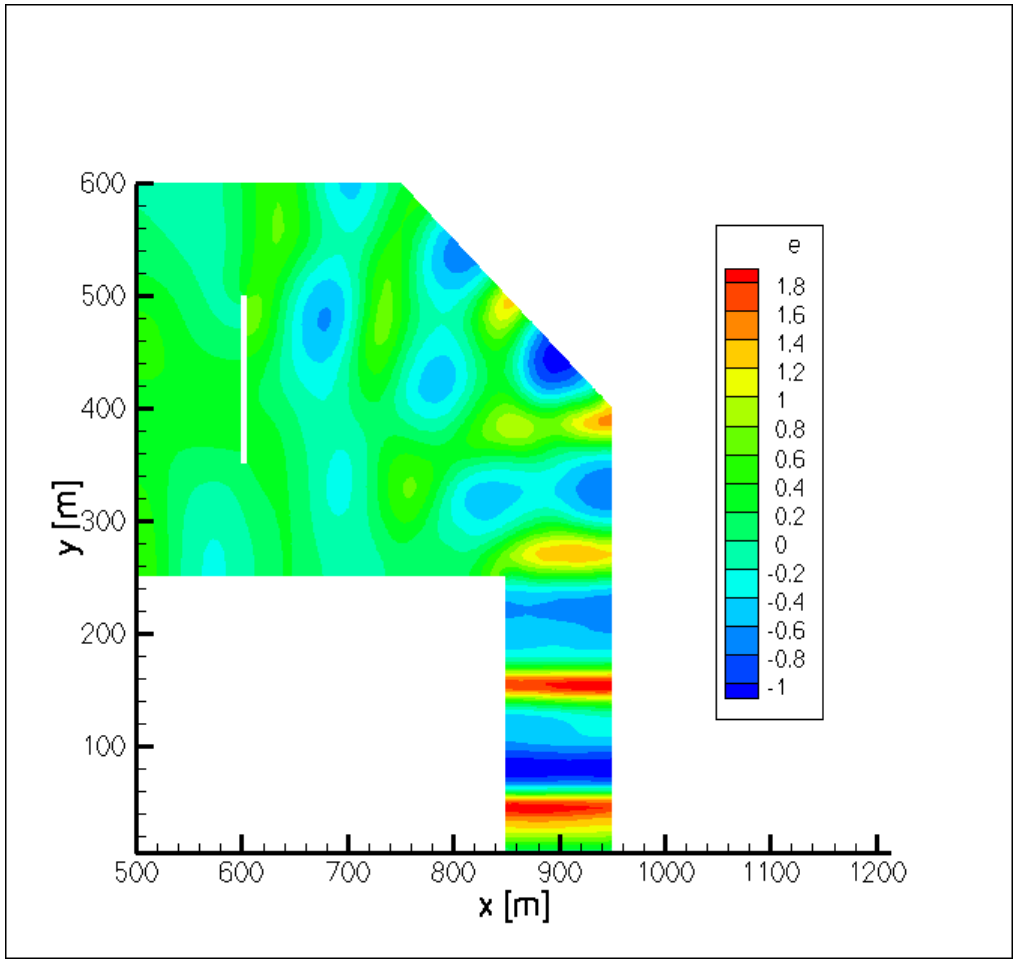


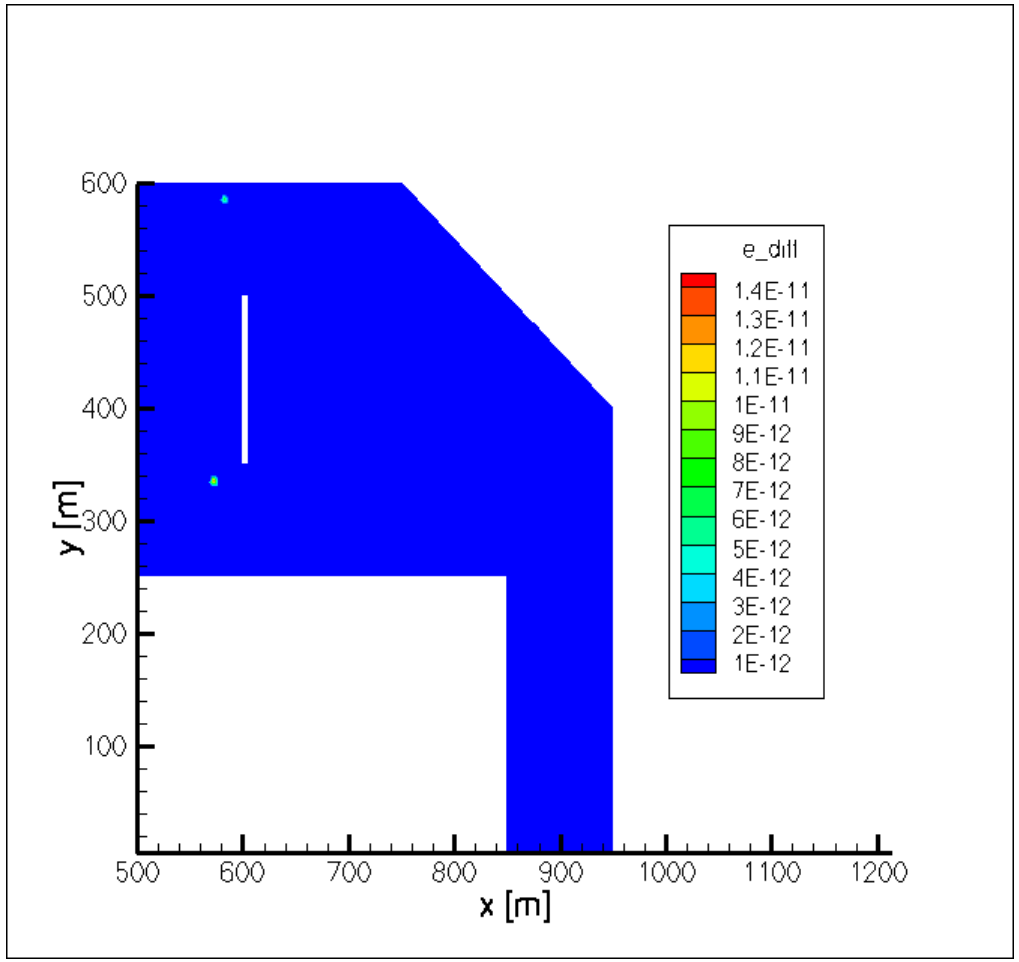
Verification and validation of parallel implementations

- **Verification**: showing that the results obtained with the serial and the parallel code (and then with different number of partitions) are *identical*
- If not identical: why the differences occur?
- Formulate recommendations

- **Validation**: showing that applying more processors brings a speedup and the code efficiency is not affected – *scalability*
- If not: why the efficiency differs?
- Formulate recommendations

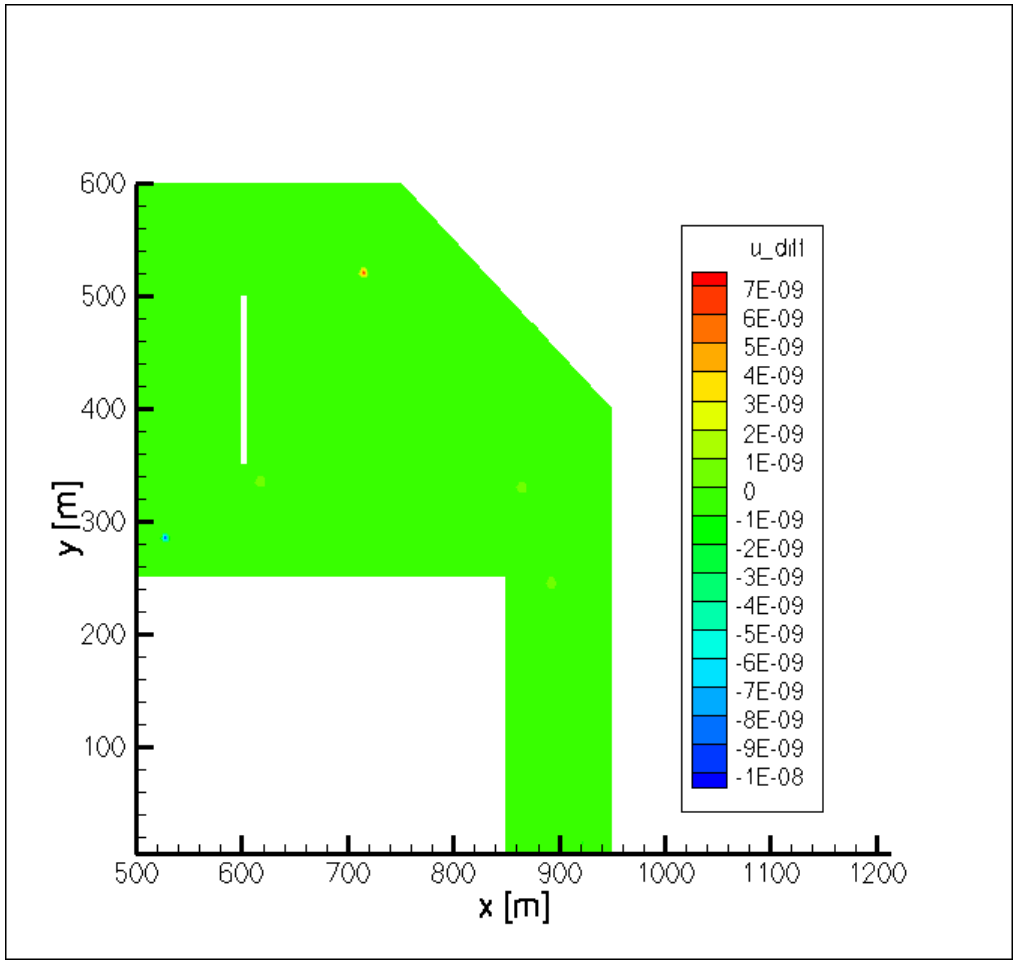
Harbour test case

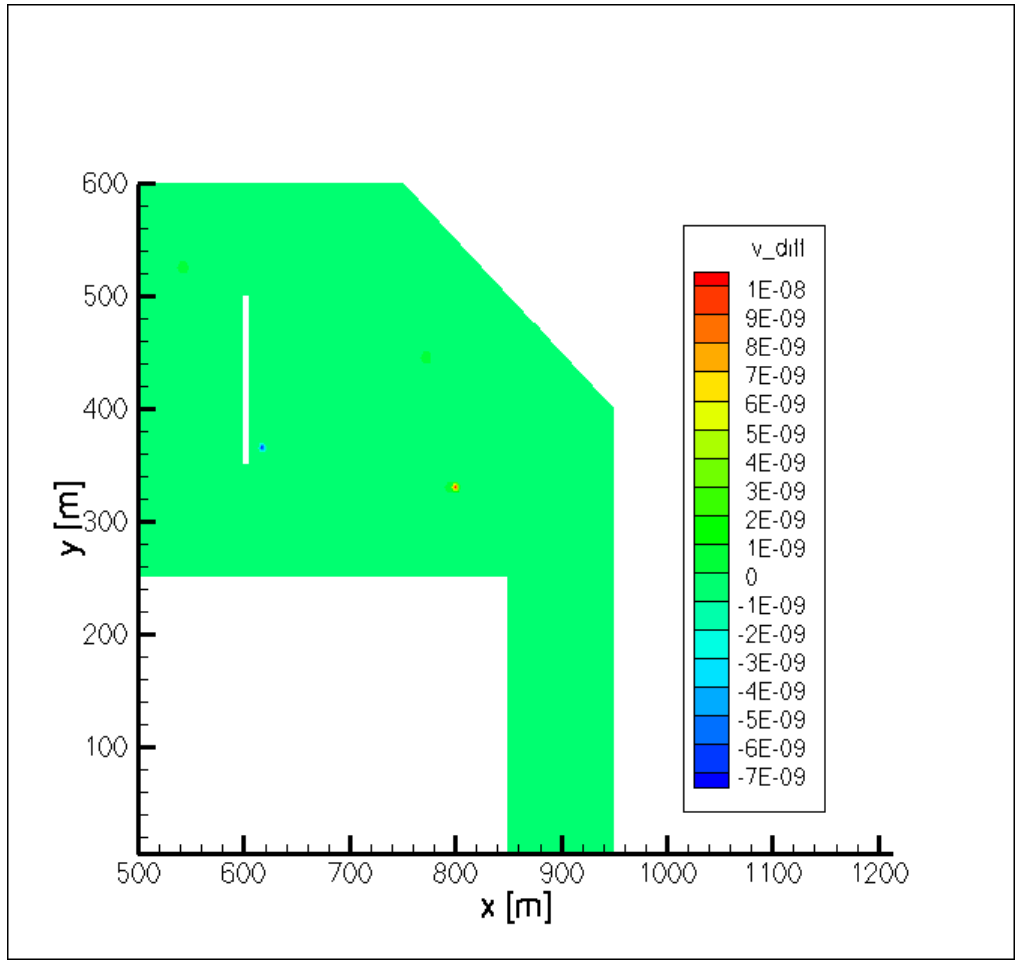




A regular mesh of triangles, flat bottom, short waves

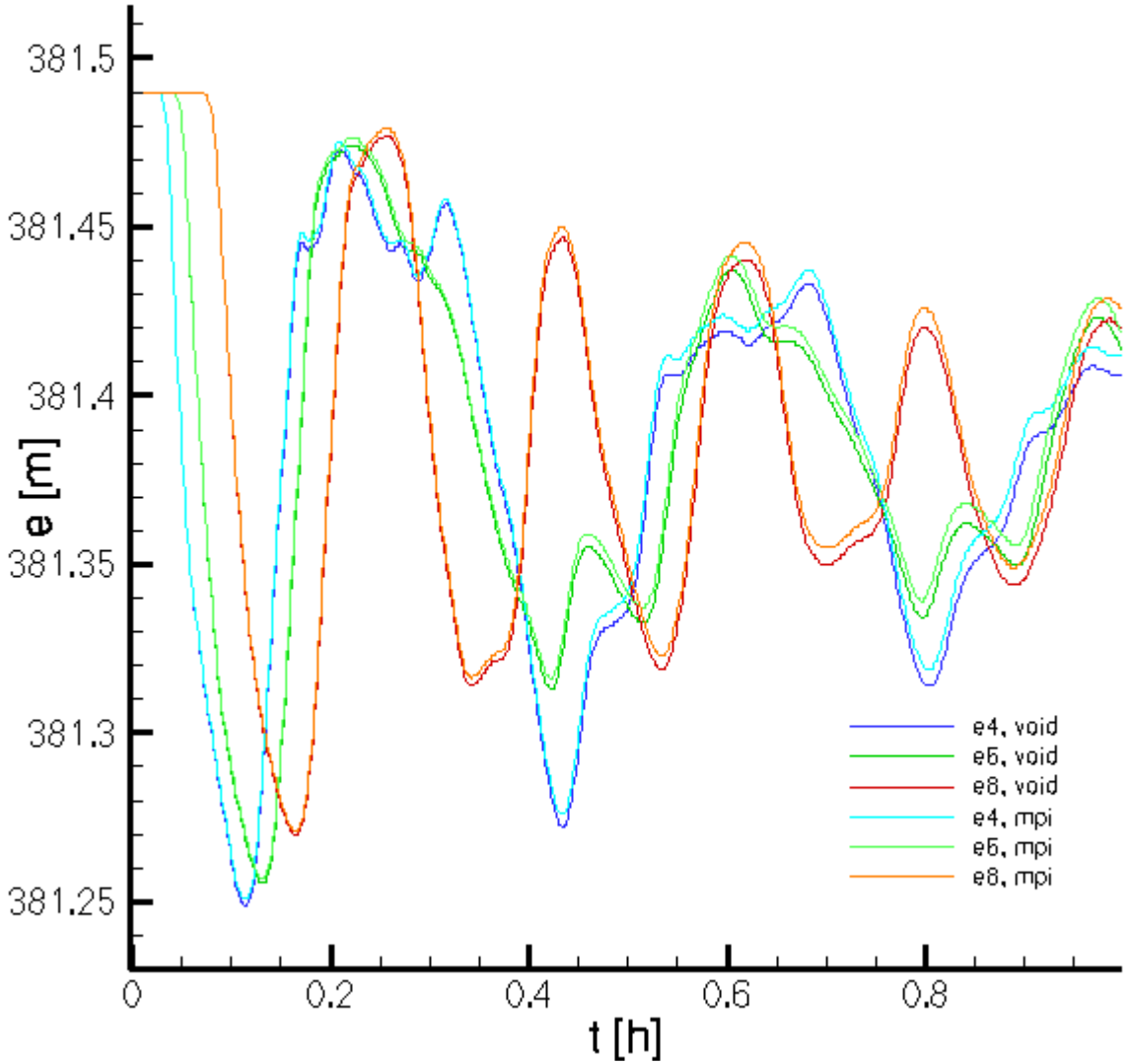
Harbour test case





A completely non-hydrostatic test case

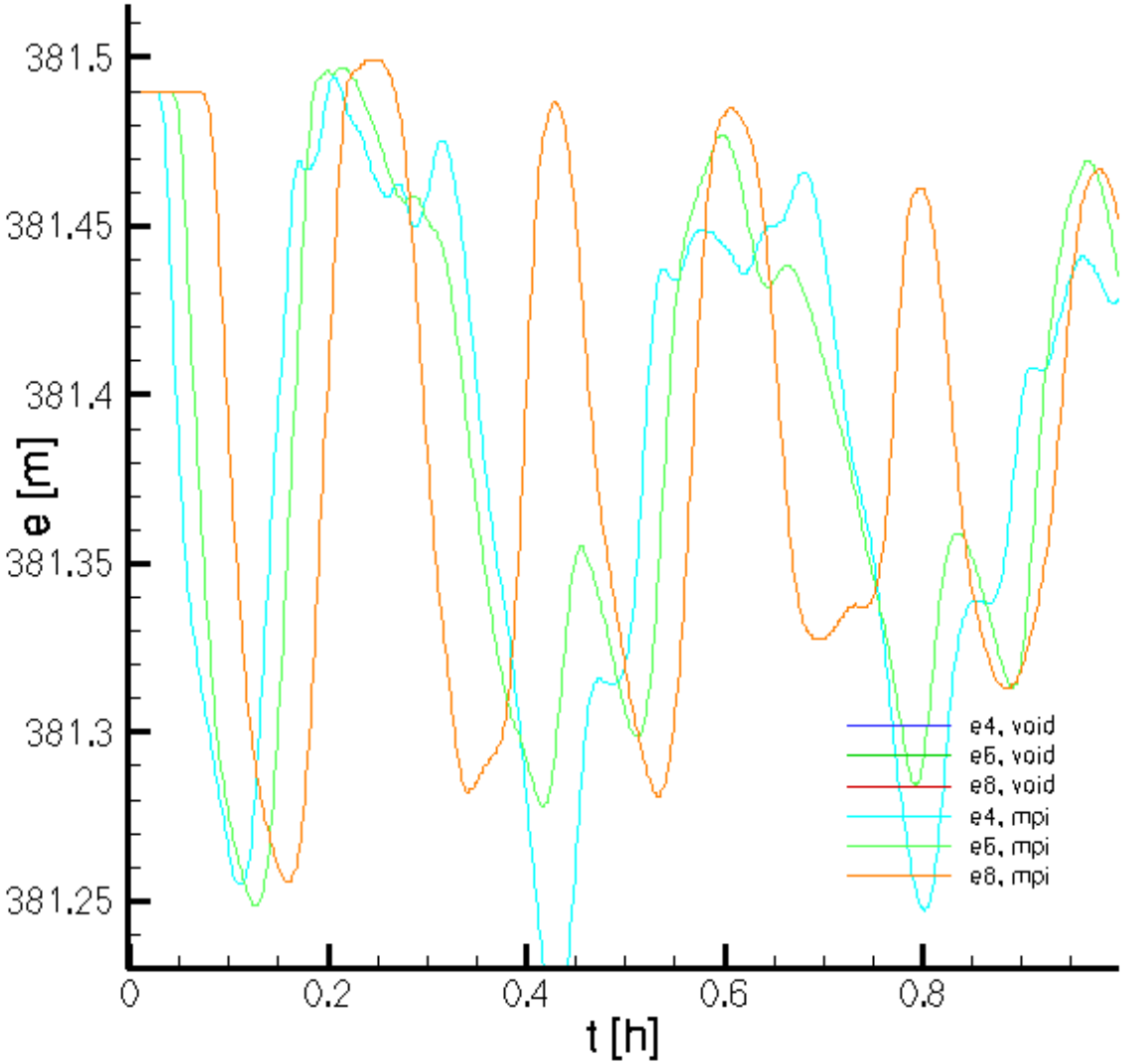
Lock waves in a channel



Waves induced in a channel stretch due to lock operation

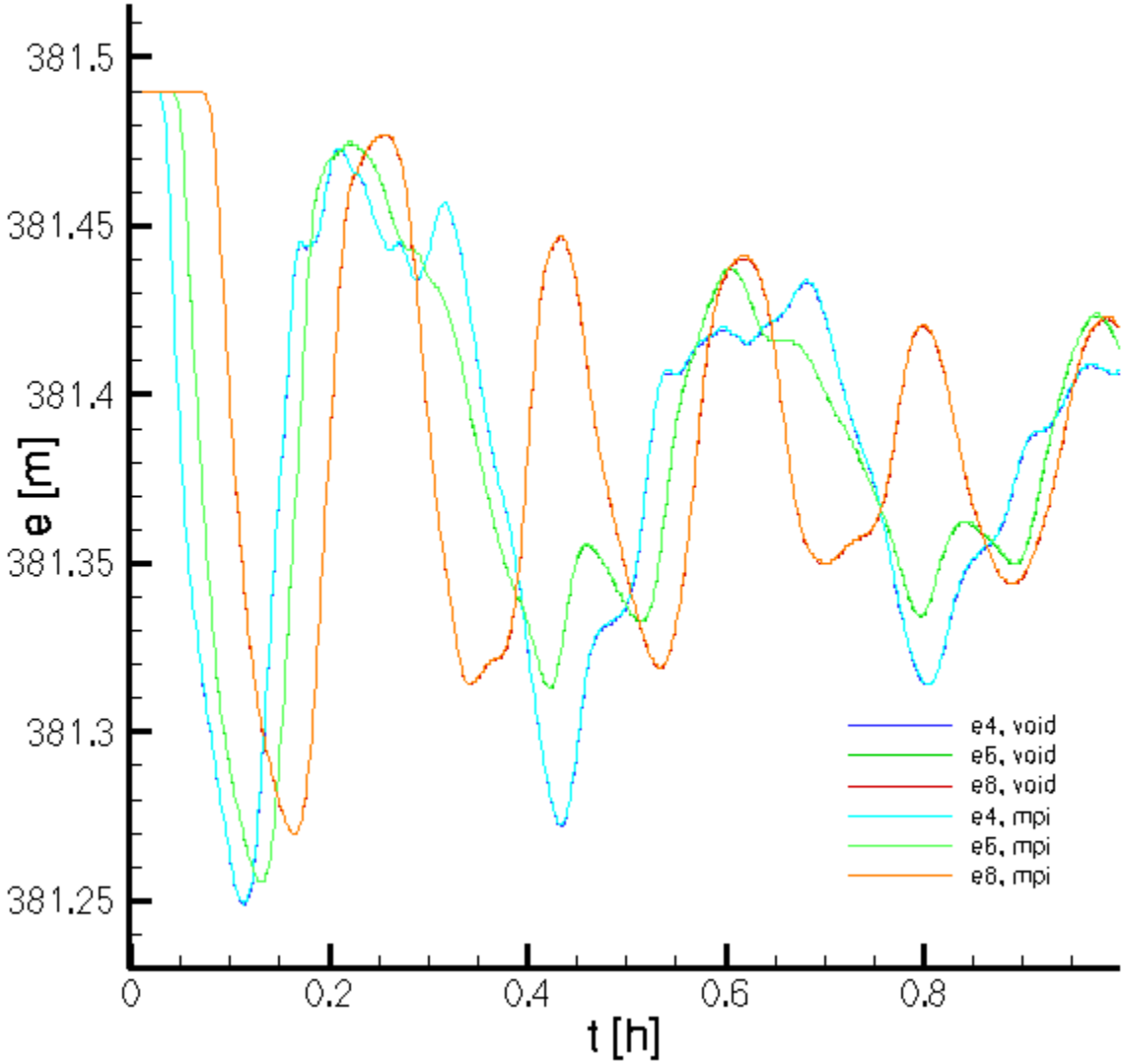
Elevation curves for one and four processors each show some systematical error and artificial diffusion…

Lock waves in a channel



Without advection, elevation curves for one and four processors cover each other

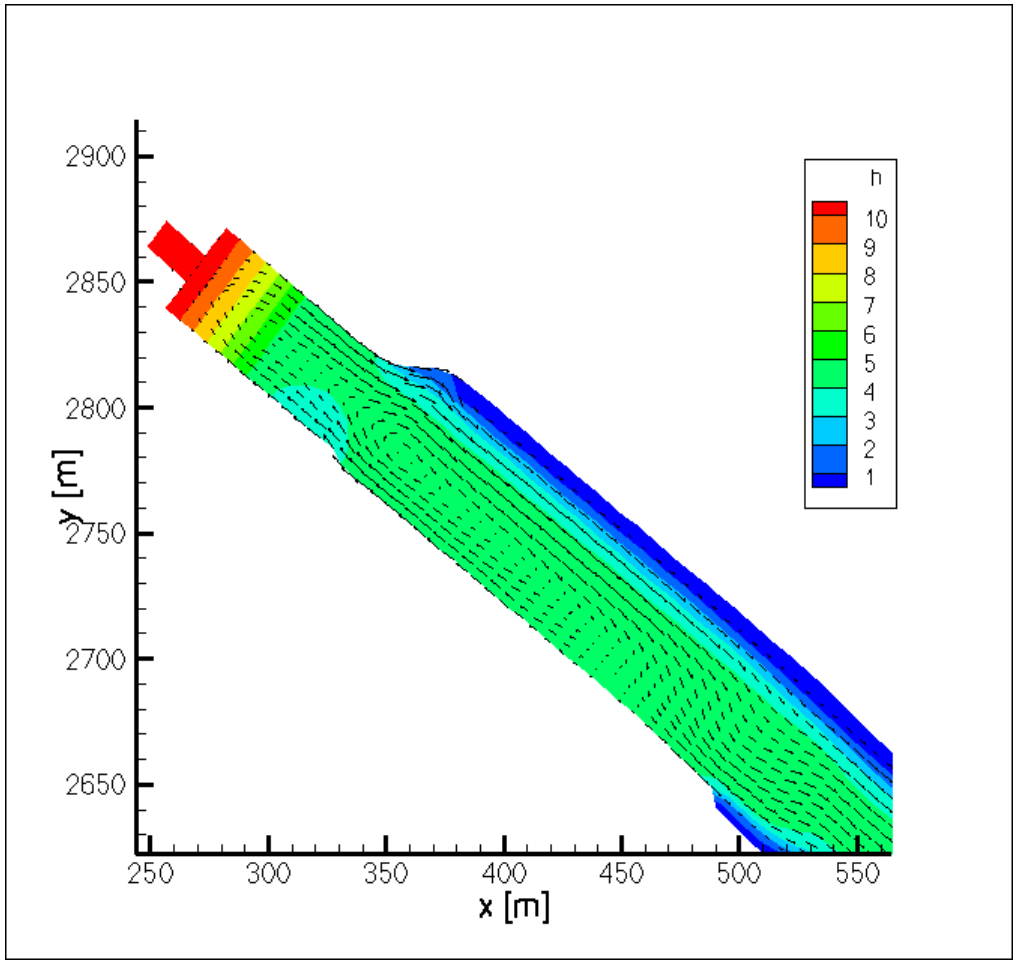
Lock waves in a channel

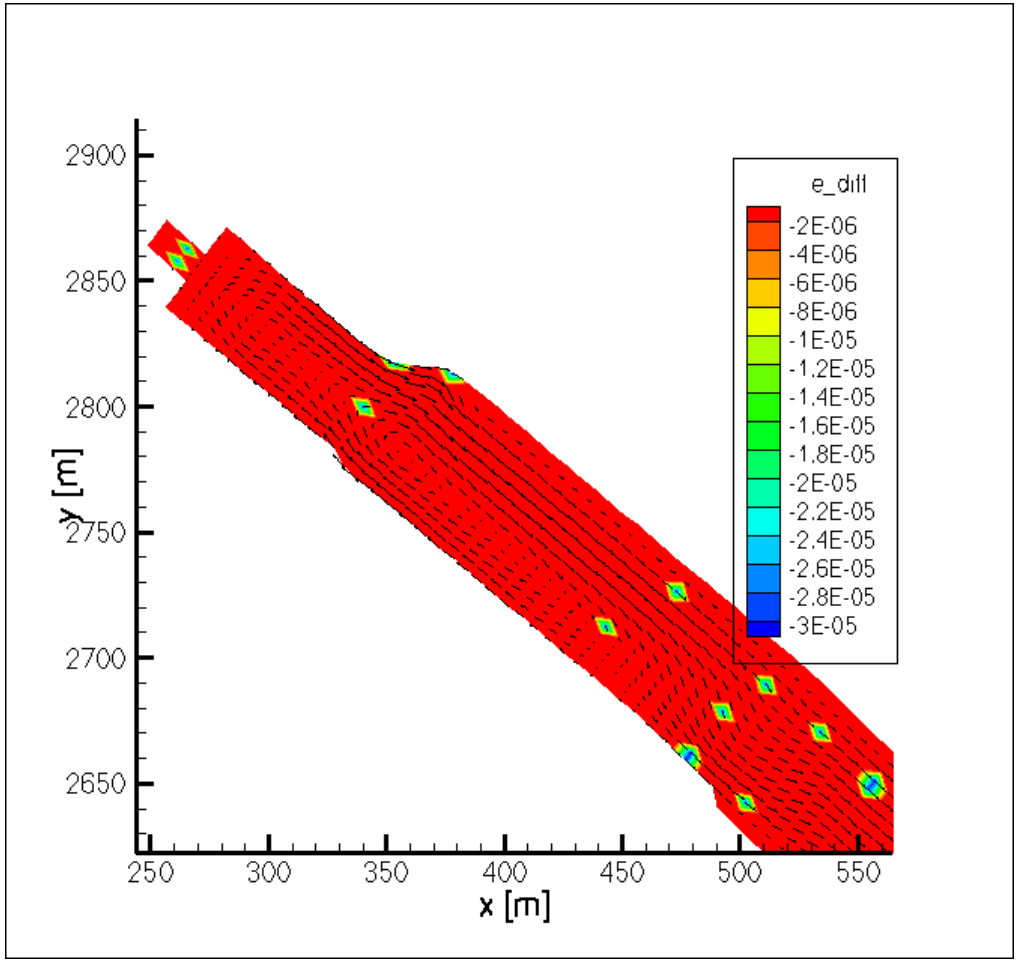


Inserting one call for computing the mean value of velocity on subdomain interfaces causes the elevation curves to (almost) cover each other again

The agreement even better if the **halo depth is increased**

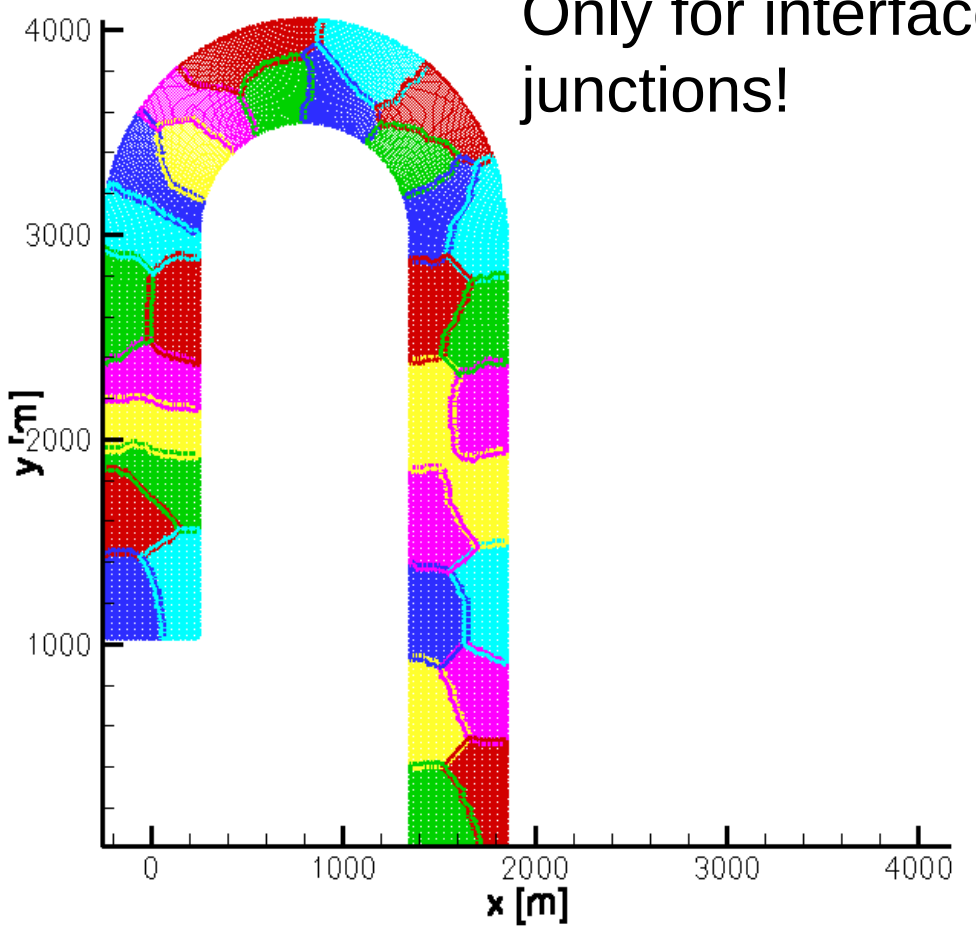
Lock waves in a channel

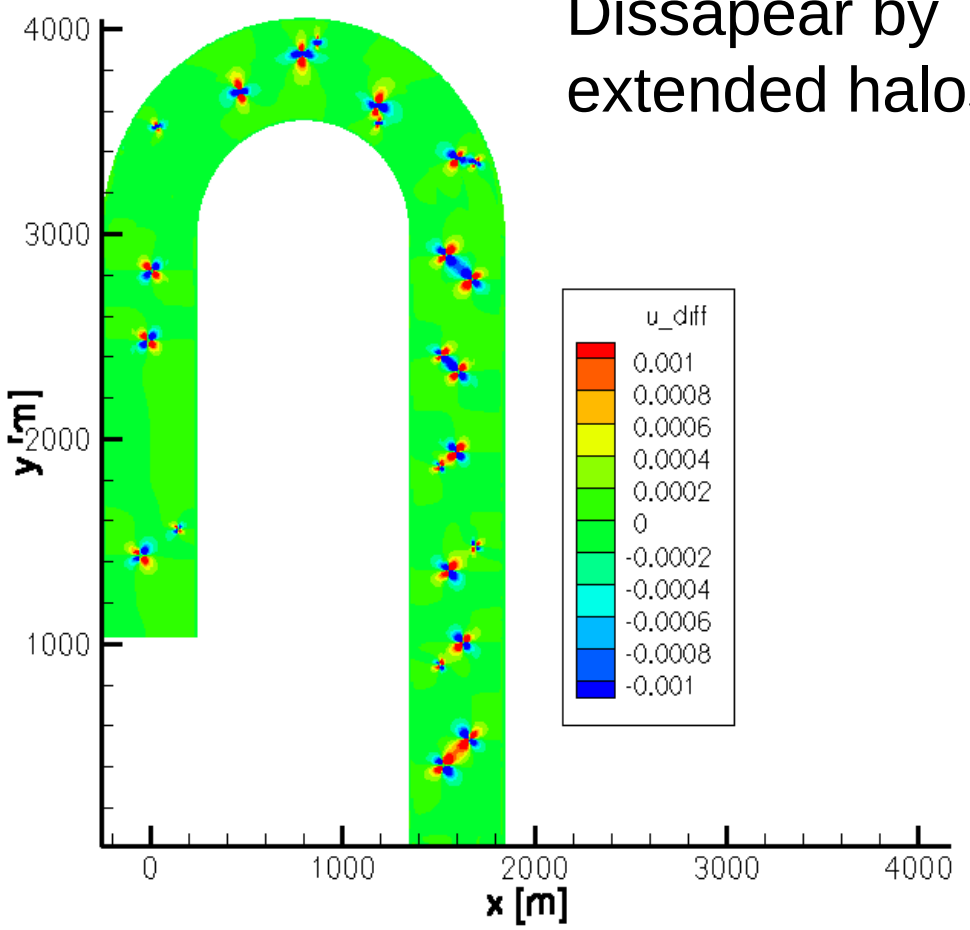




A very regular mesh, quadrangles

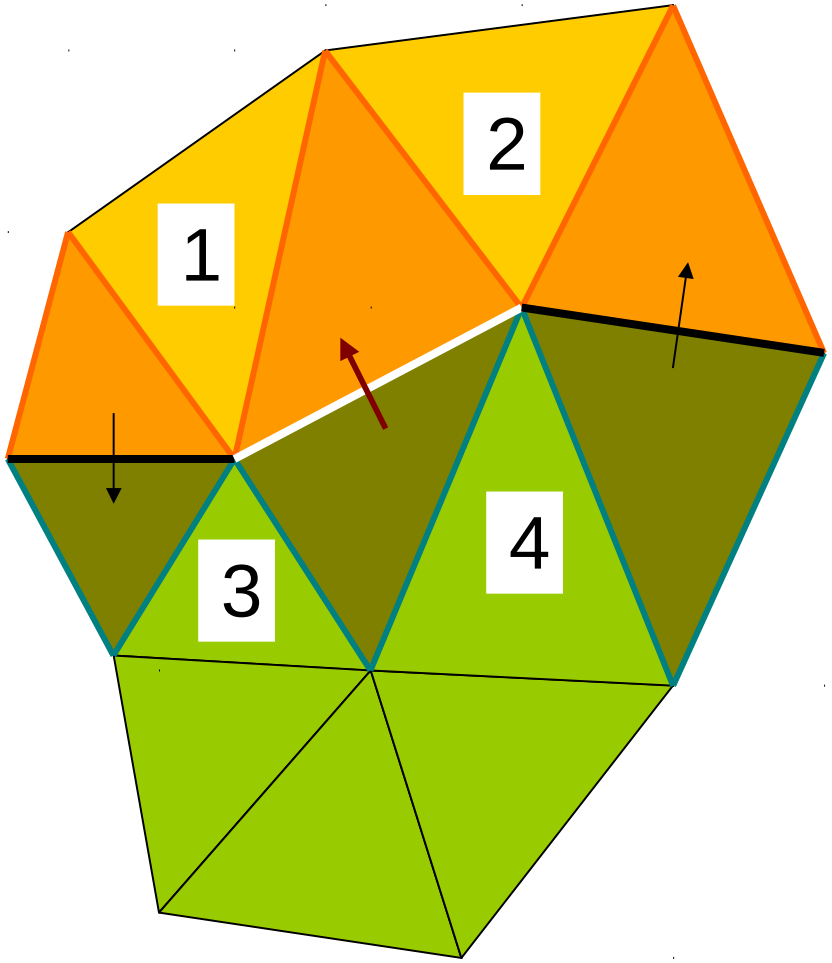
Disagreement in velocity, serial versus parallel





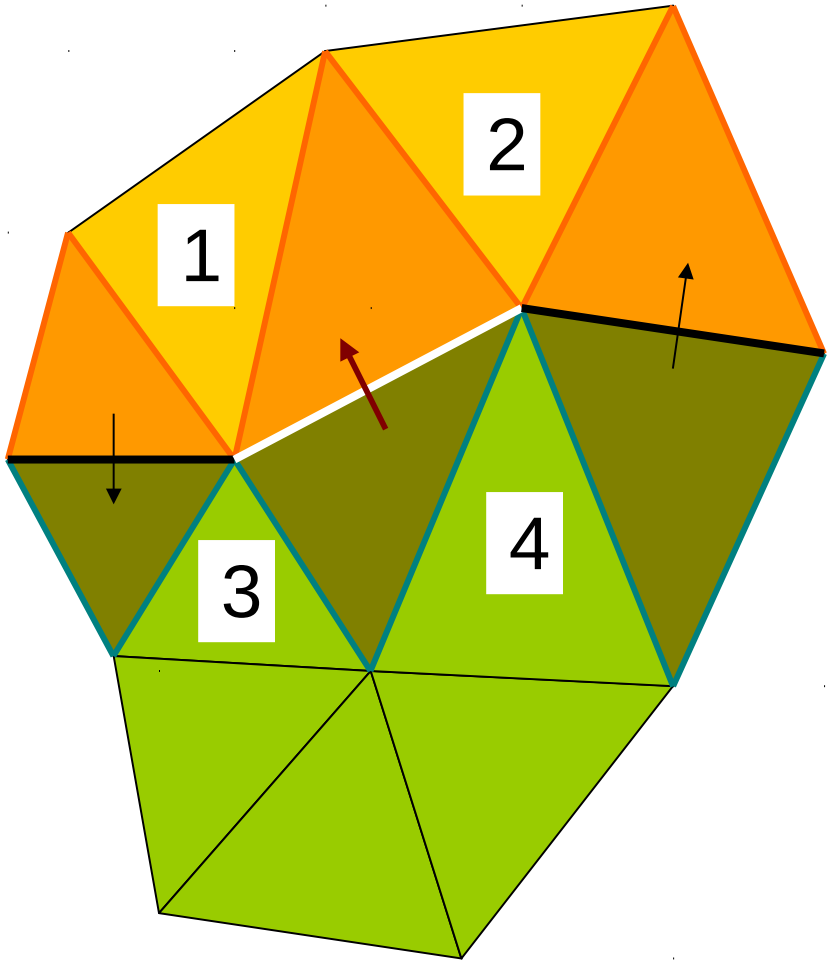
Diffusion neighbours

- Computing “diffused” velocity requires values not only from the neighbour cells of a face, but also…
- …from **four** further *viscosity neigbours* of these cells

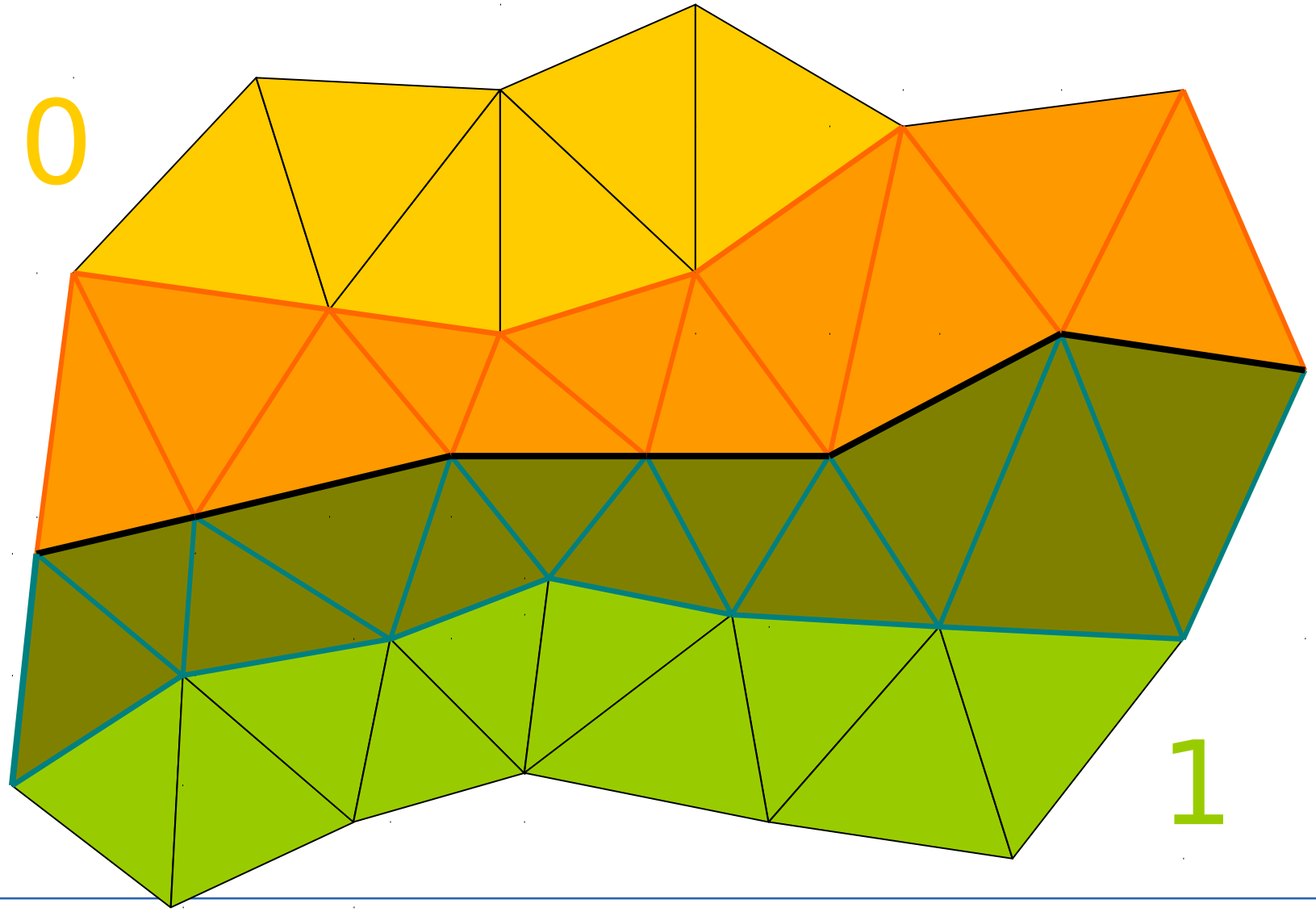


Diffusion neighbours

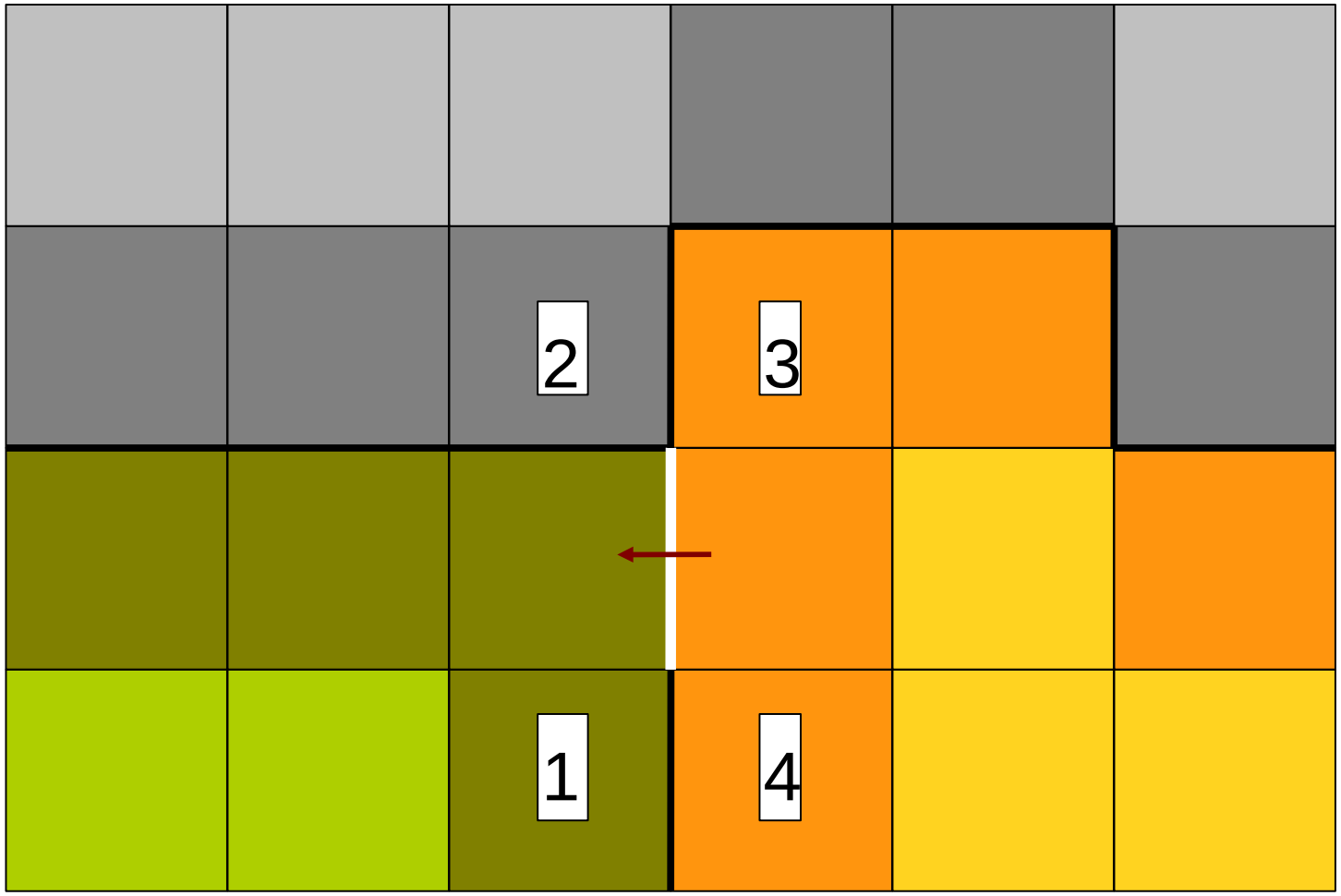
- Yellow neighbour yields contributions from own cells 1 and 2
- Green neighbour yields contributions from own cells 3 and 4
- They are summed up via – *assembling*



Extended halos (one dependency further on)

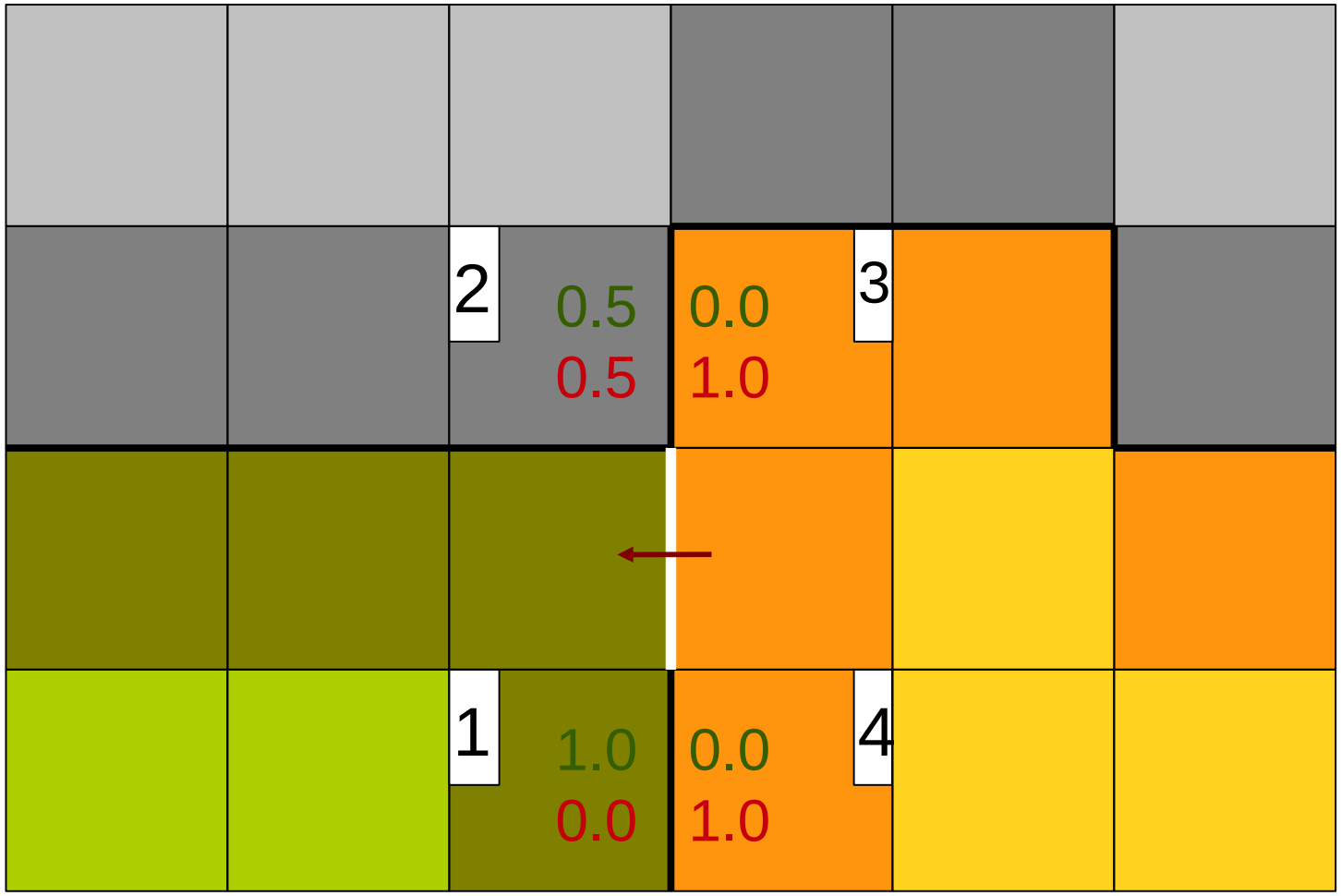


Diffusion operator



- Computing “diffused” velocity requires assembling values from four cells in a pattern

Diffusion operator



Deliver a part of the contribution if the cells are shared between partitions

The sum of the contribution factors from both neighbouring partitions must be one.

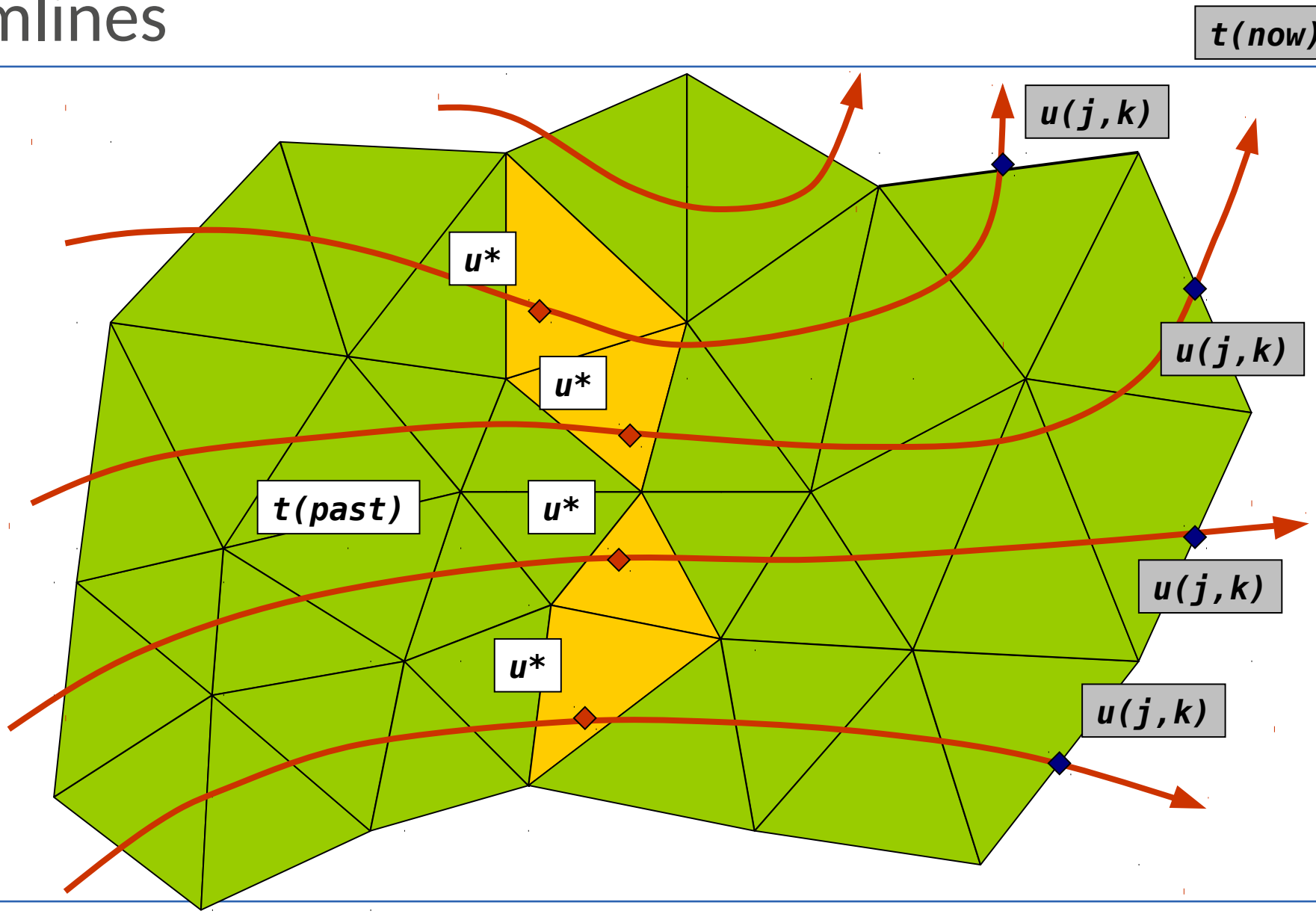
“weighted assembling”

$n_1=0.5c_1 + 1.0*c_2 + 0.0*c_3 + 0.5*c_4$

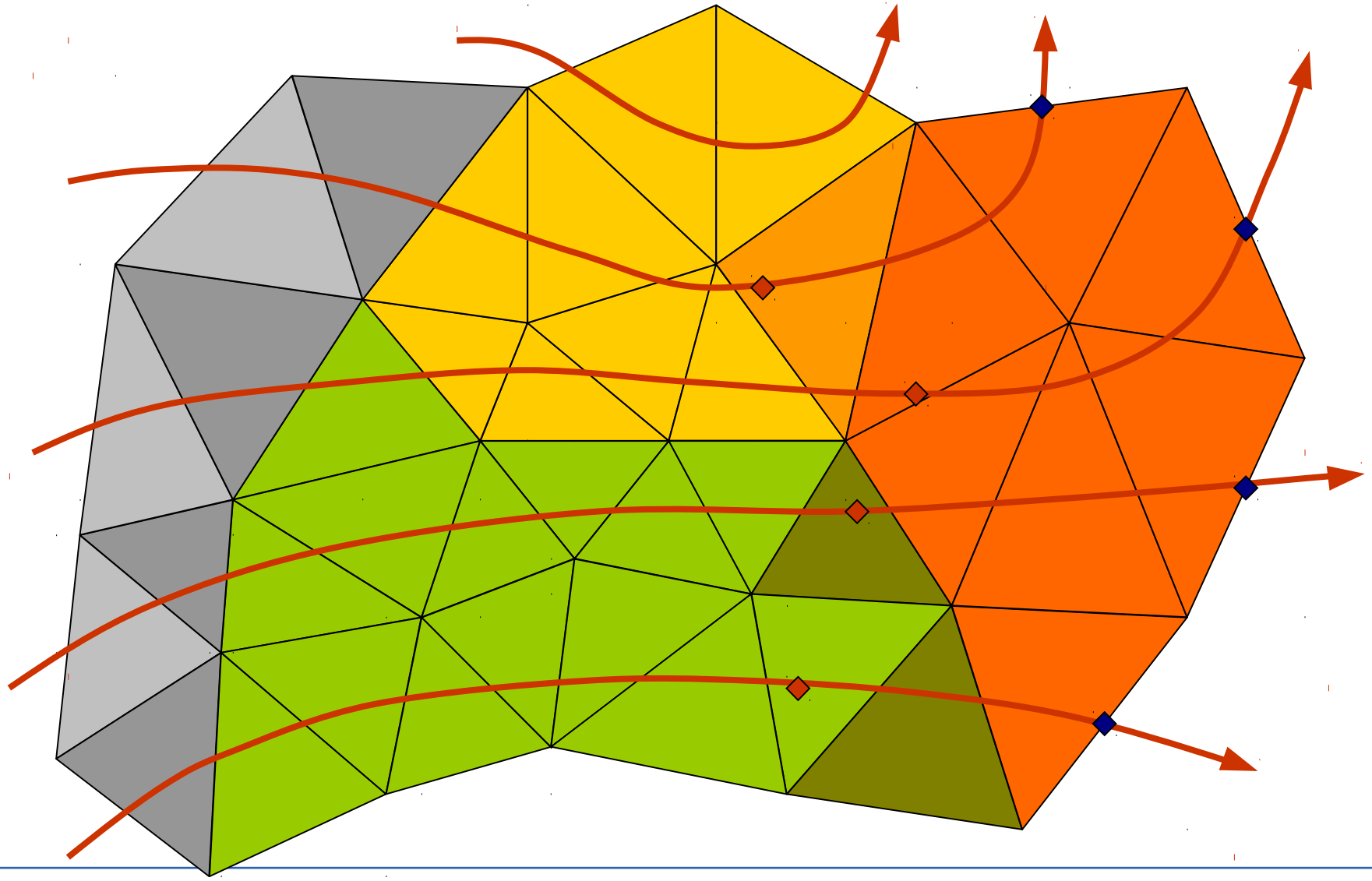
$n_2=0.5c_1 + 0.0*c_2 + 1.0*c_3 + 0.5*c_4$

$n = a + n_1 + n_2$

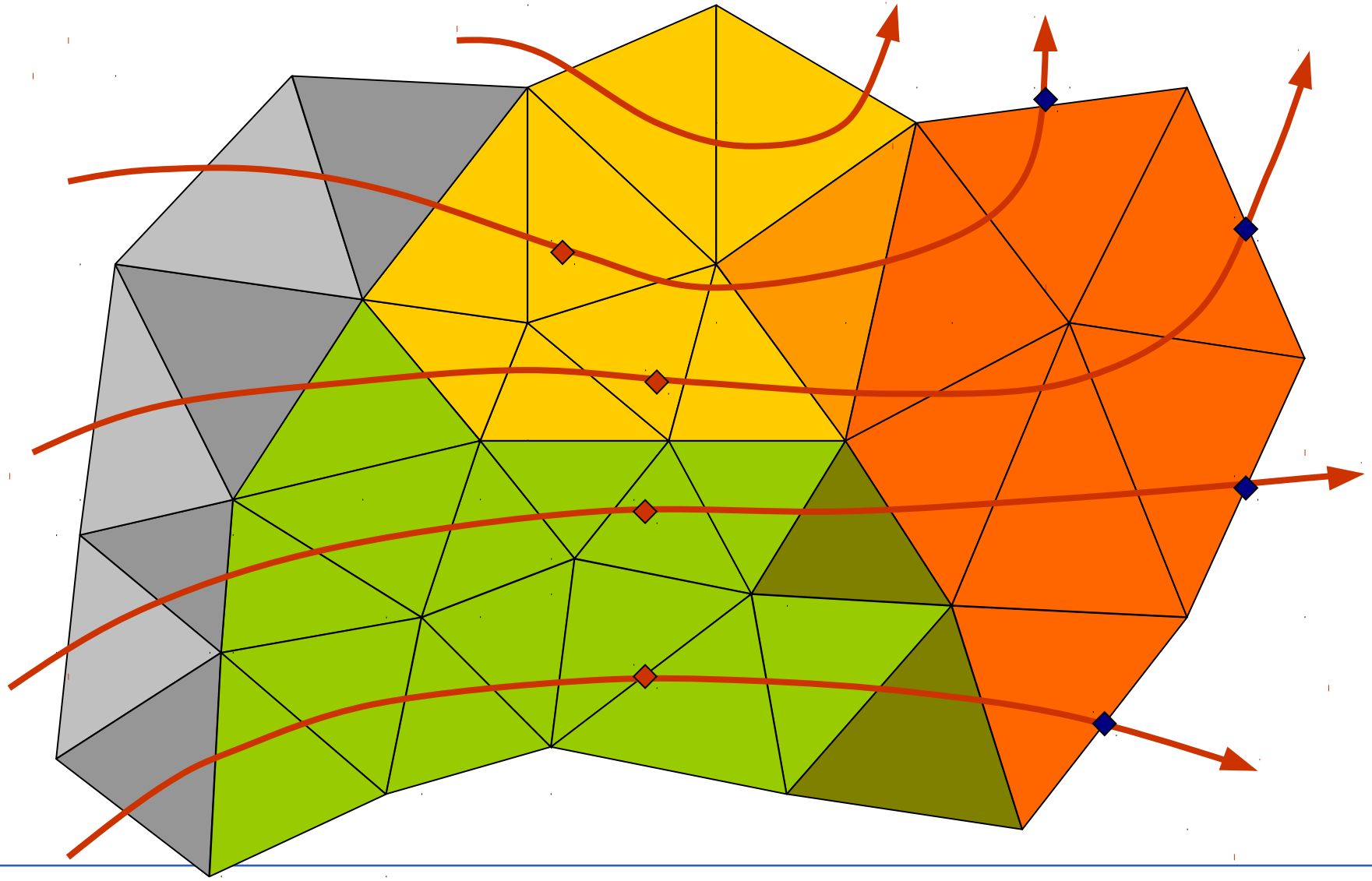
Streamlines



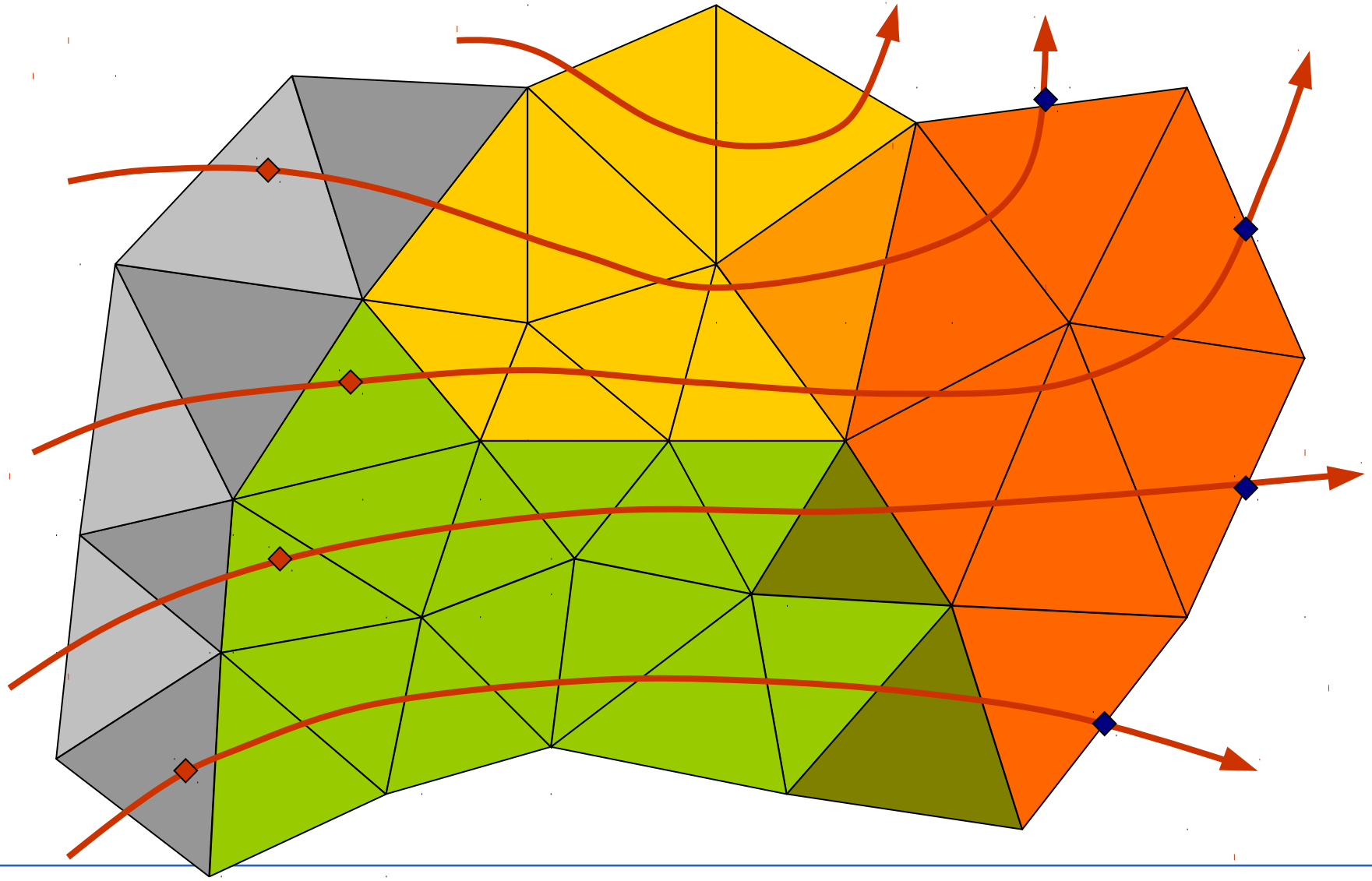
Streamline for tracking Cr >1



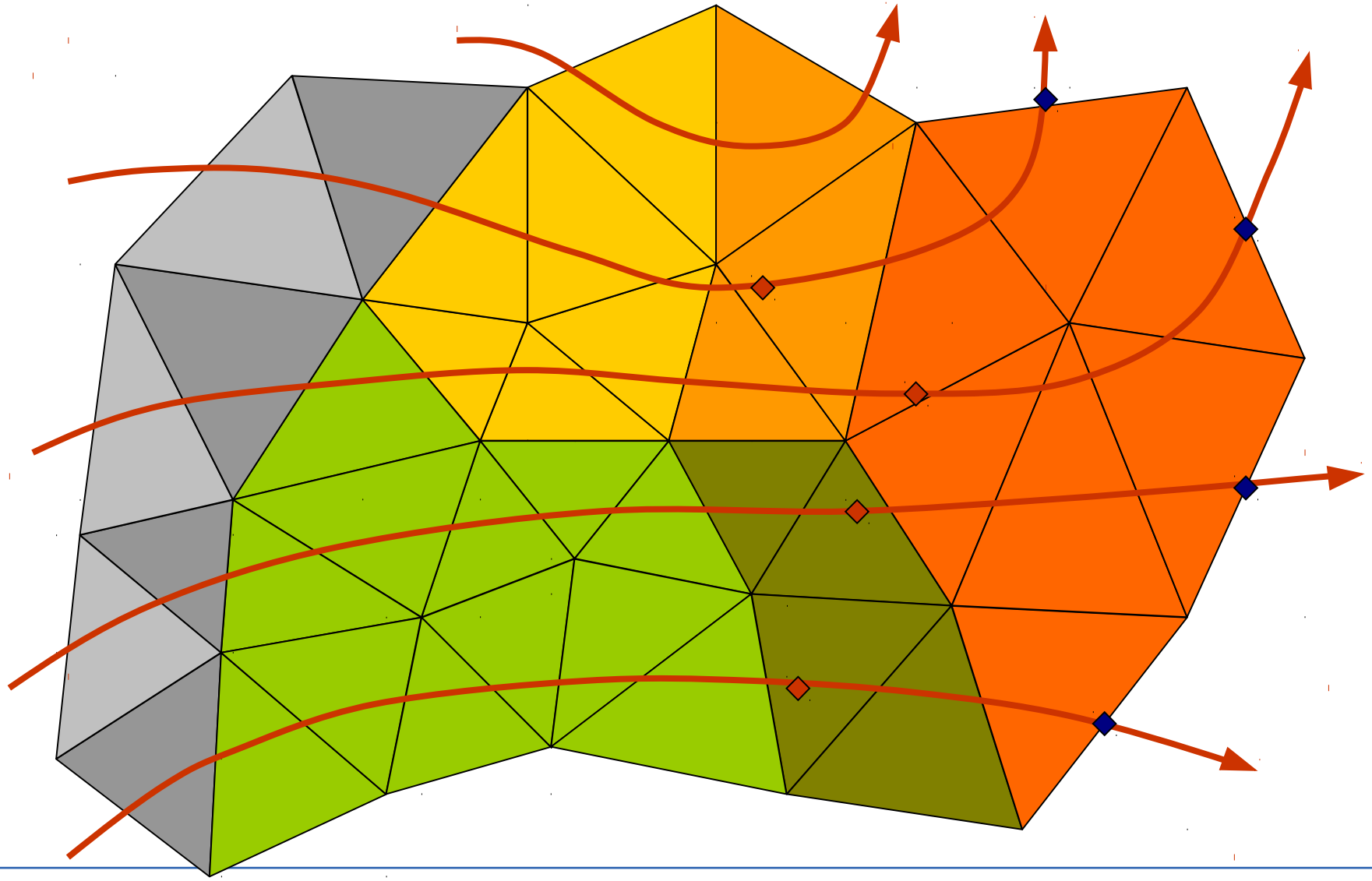
Streamline tracking for Cr >> 1



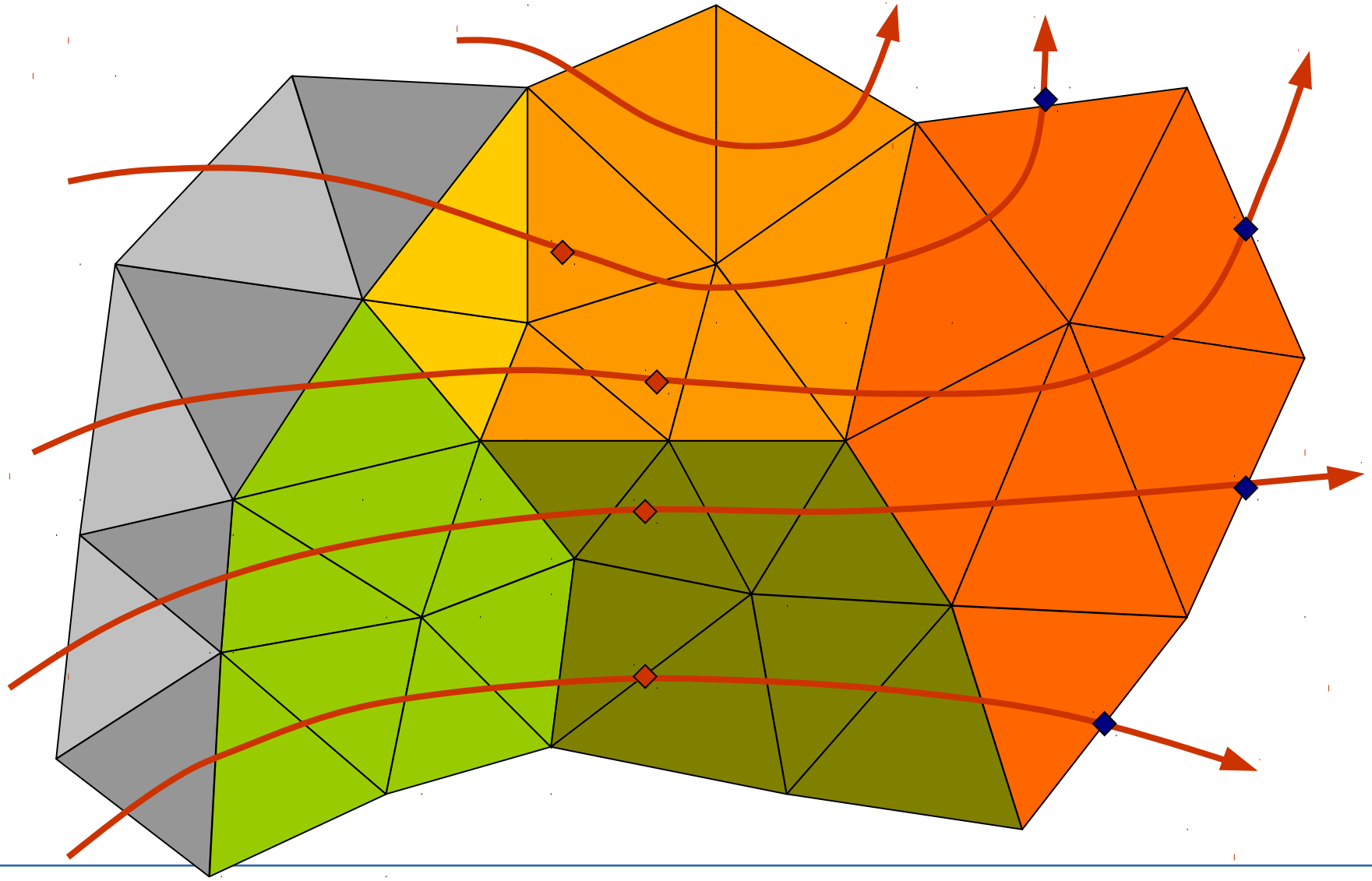
Streamline tracking for Cr >>> 1



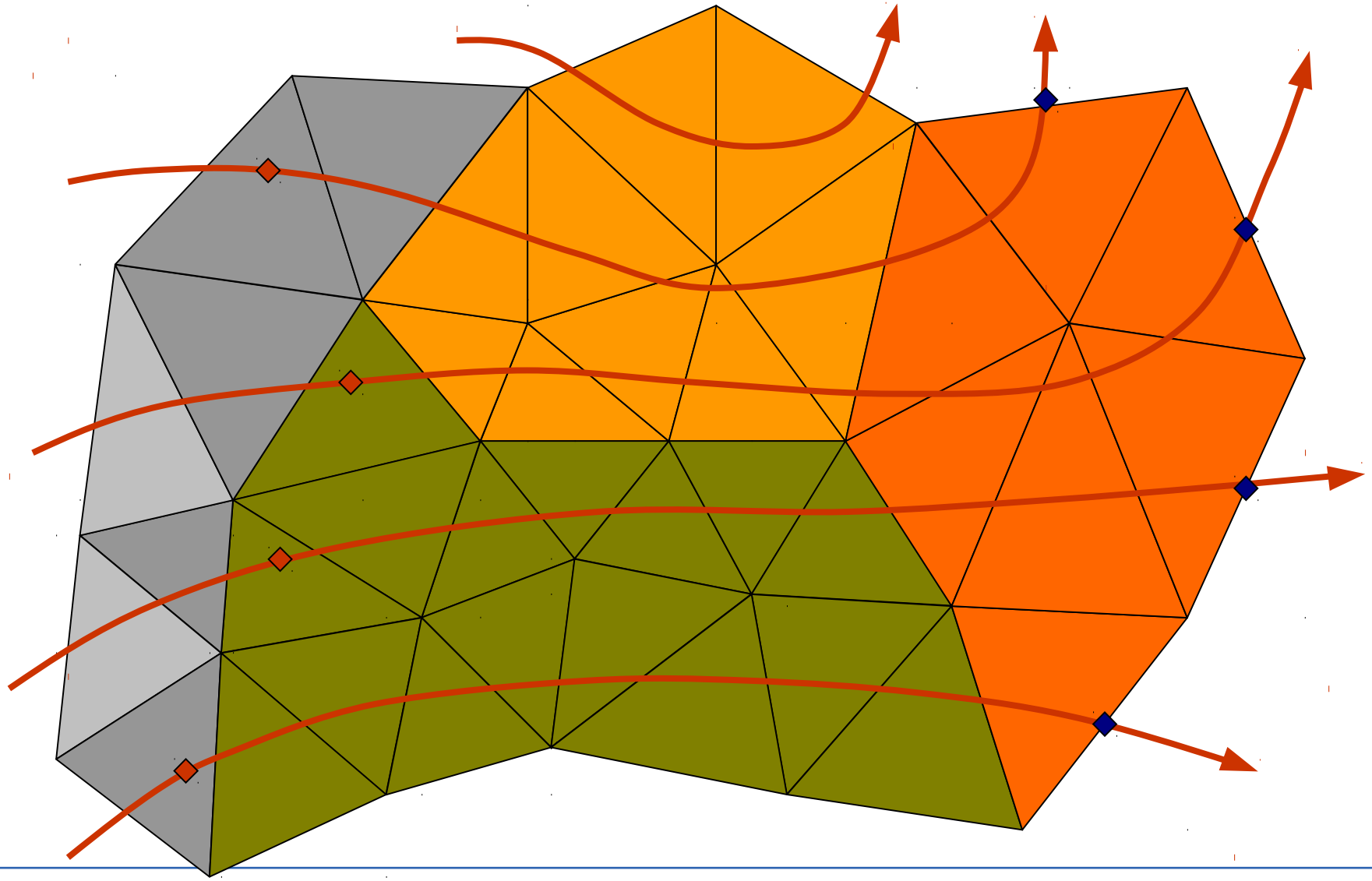
Extend halos?



Extend halos??



Extend halos???



Dog tags for lost trackbacks

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
```

Source: dogtagshop.de

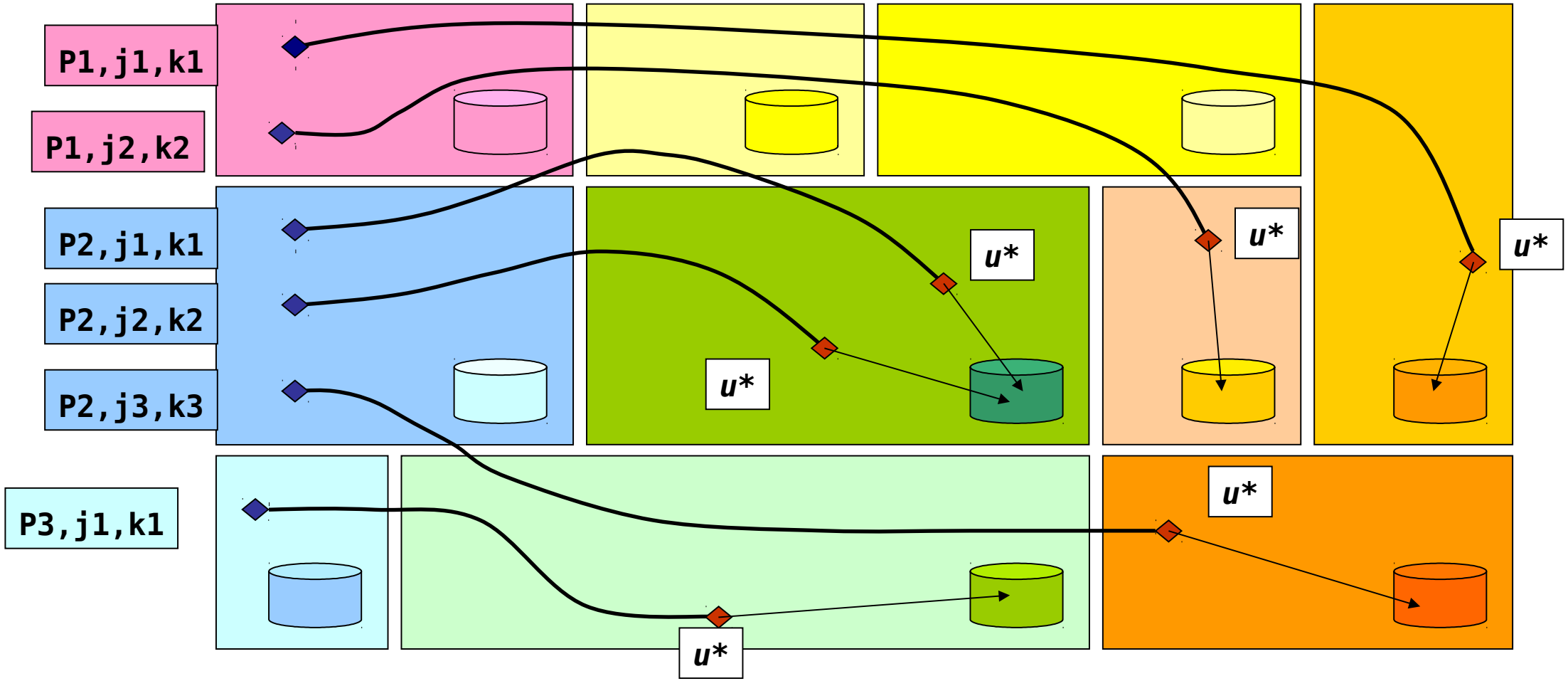
Autonomous tracking

- 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`**

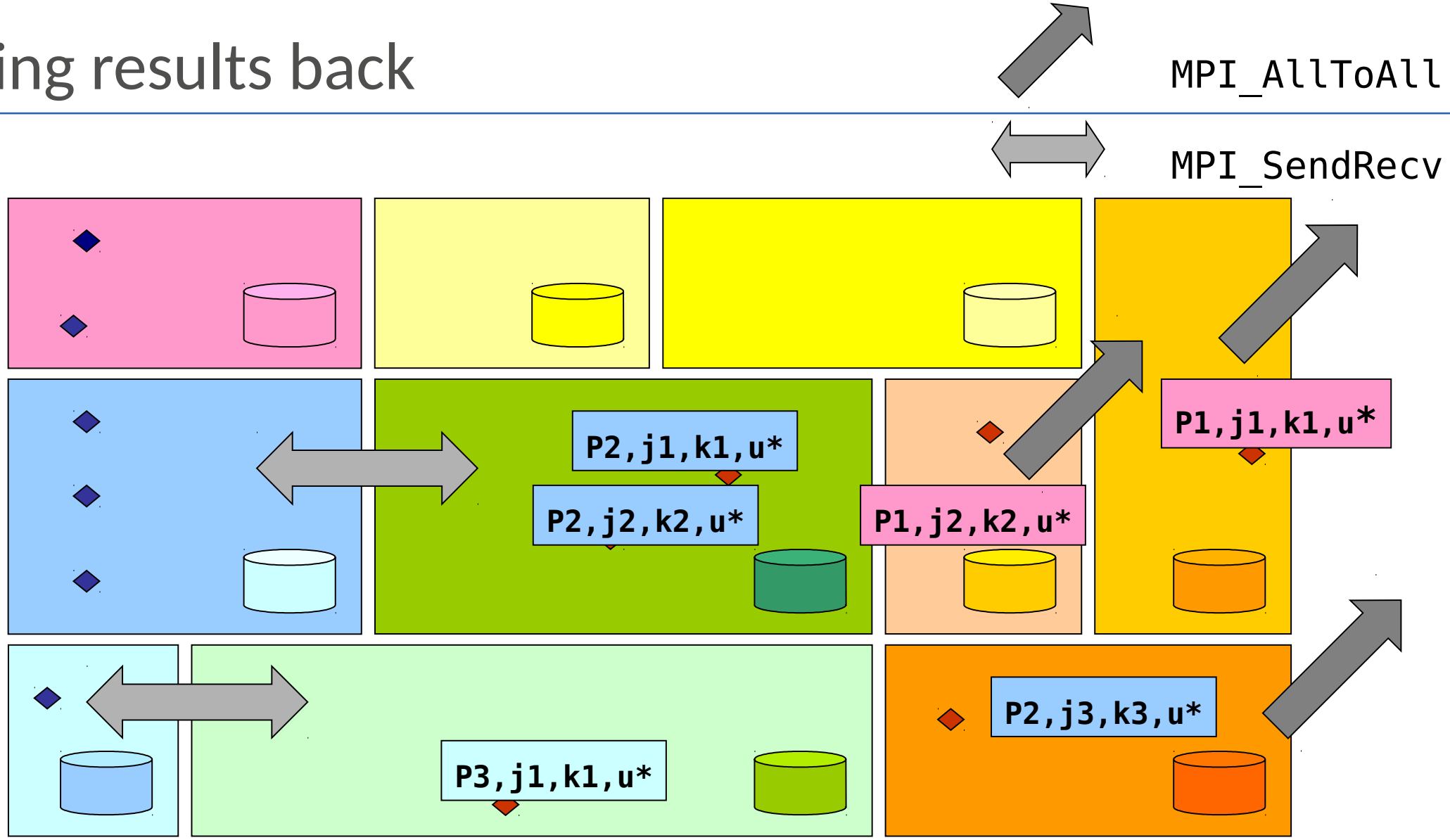
Autonomous tracking

- 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

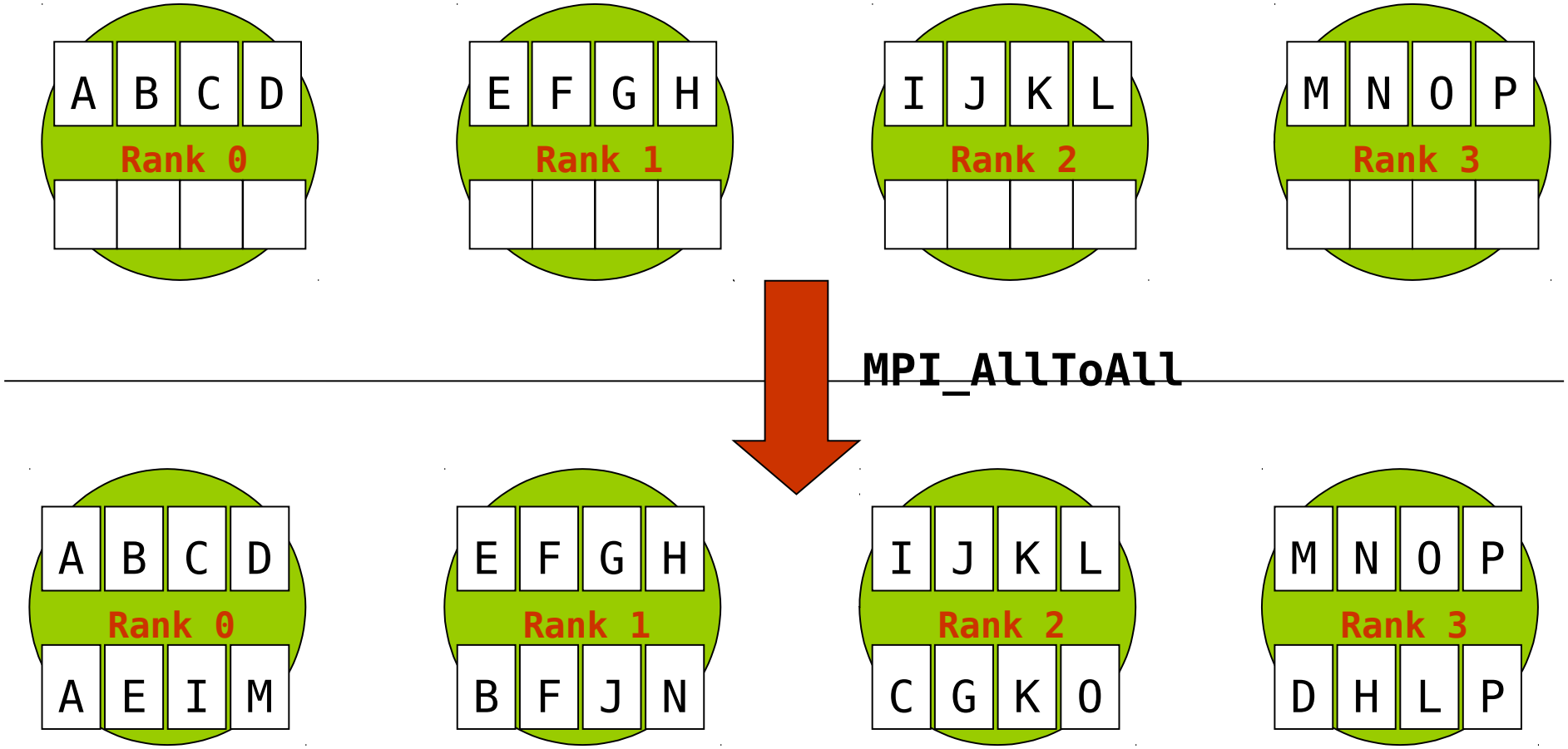
Autonomous tracking



Sending results back



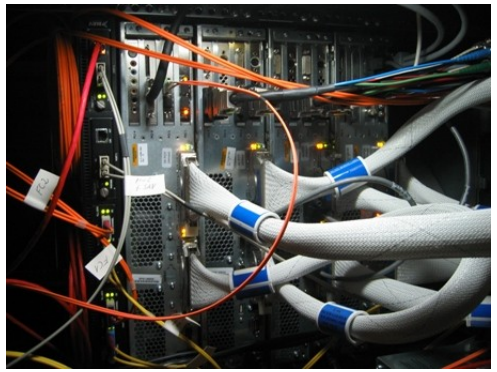
MPI_AllToAll



Summary: communication im MPI-UnTRIM

- FD/FV (Eulerian):
  - swapping halo values: point-to-point communication
  - common interface edges or faces: upstream values
  - exception: diffusion operator – weighted assembling

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



Recommendations

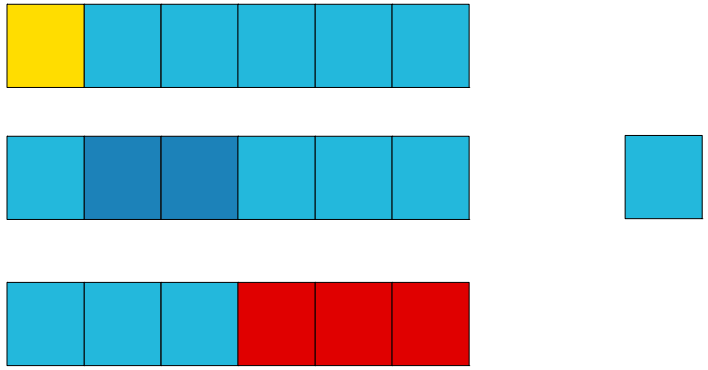
- *How to reduce differences between serial and parallel results:*
  - avoid any aspects of mesh dependencies
  - avoid over-simplifying the physics – usually by inadequate resolution
  - transport phenomena – small errors add up…

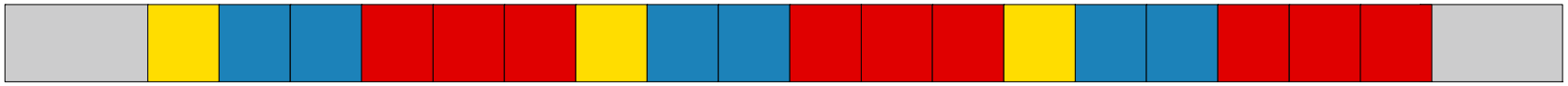
- *How to maintain the scalability – avoid:*
  - runs with more than ca. 10% halo cells
  - partitioning “in vain” - e.g. of dry-only cell areas
  - large numbers of iterations in equation solvers (>50)
  - large number of subiterations in diffusion & transport schemes (>10)
  - large number of traceback interface crossings (>5)

If there‘s still time

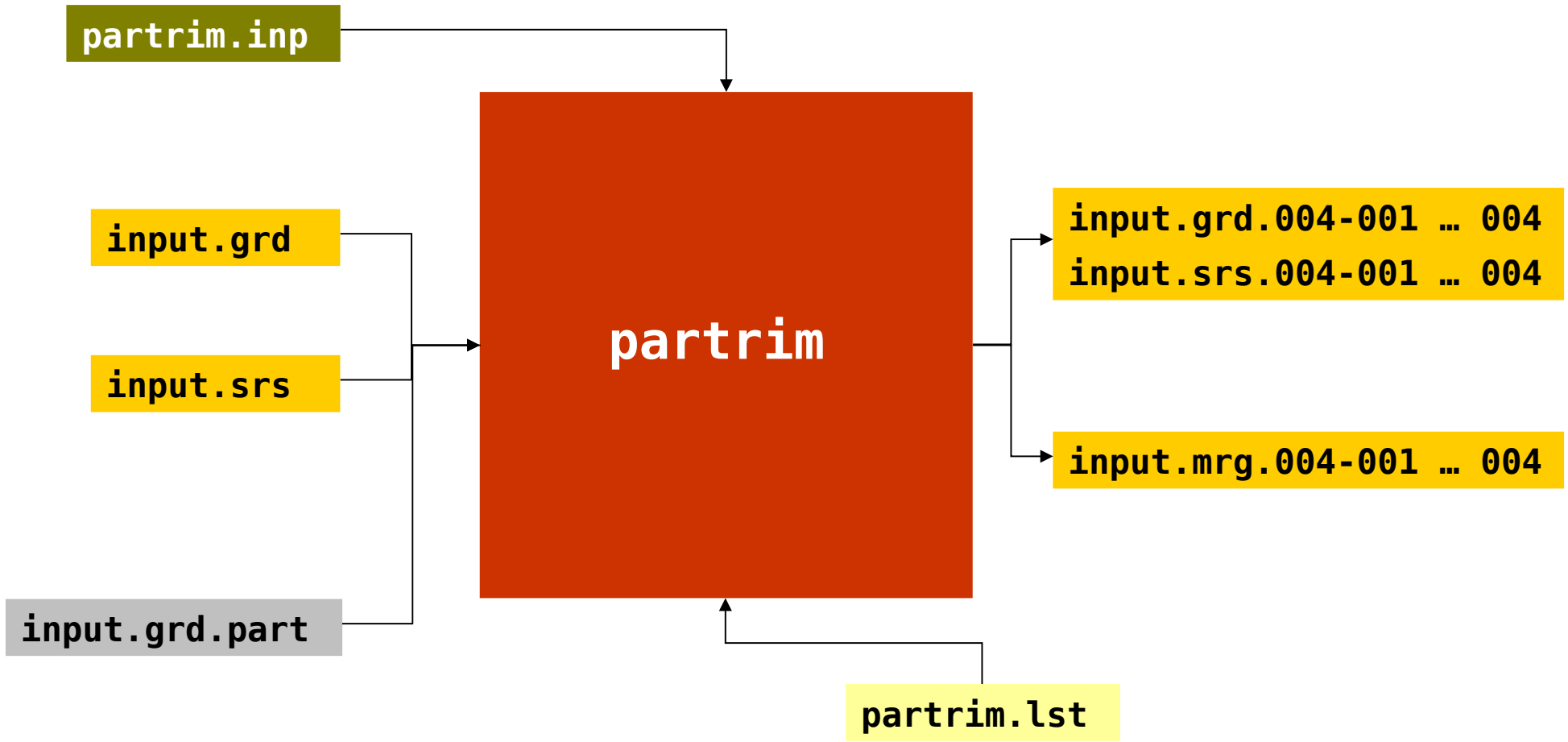
Improving and optimising UnTRIM MPI Library, 7th workshop, Trento, 3-5 May **2010**

- **Parallel I/O with MPI**
- (Improving communication itself)

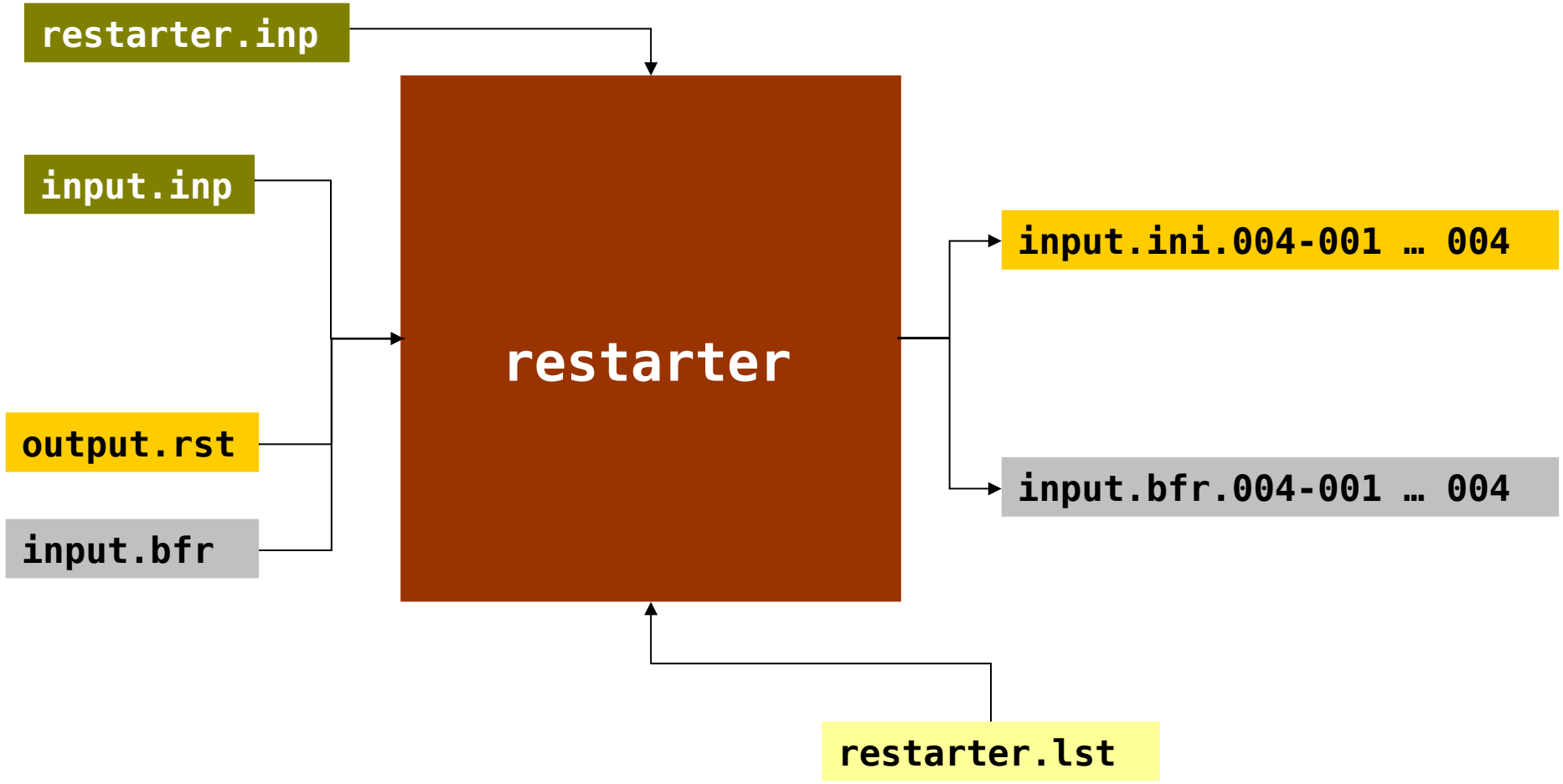




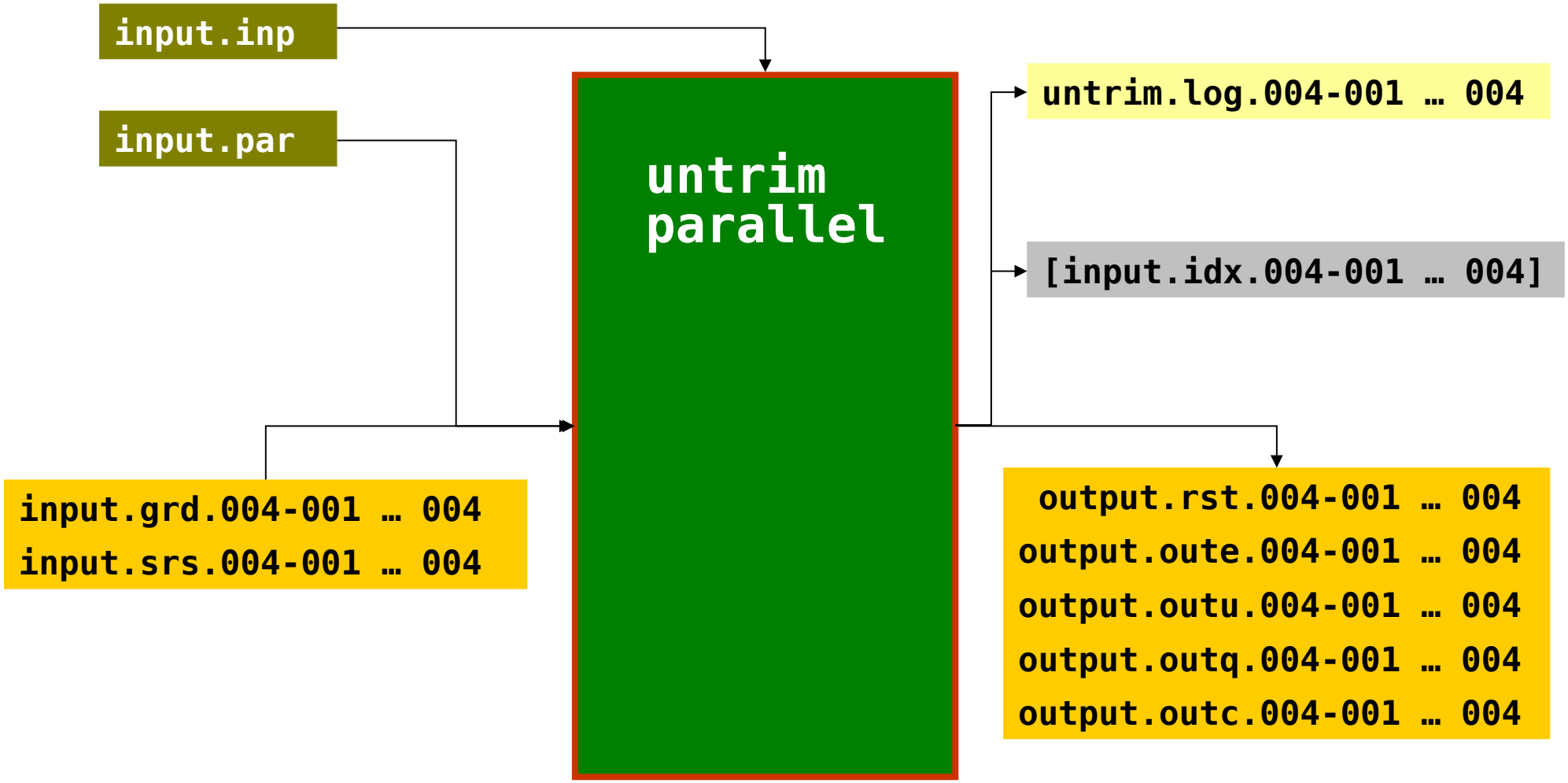
Mesh partitioning



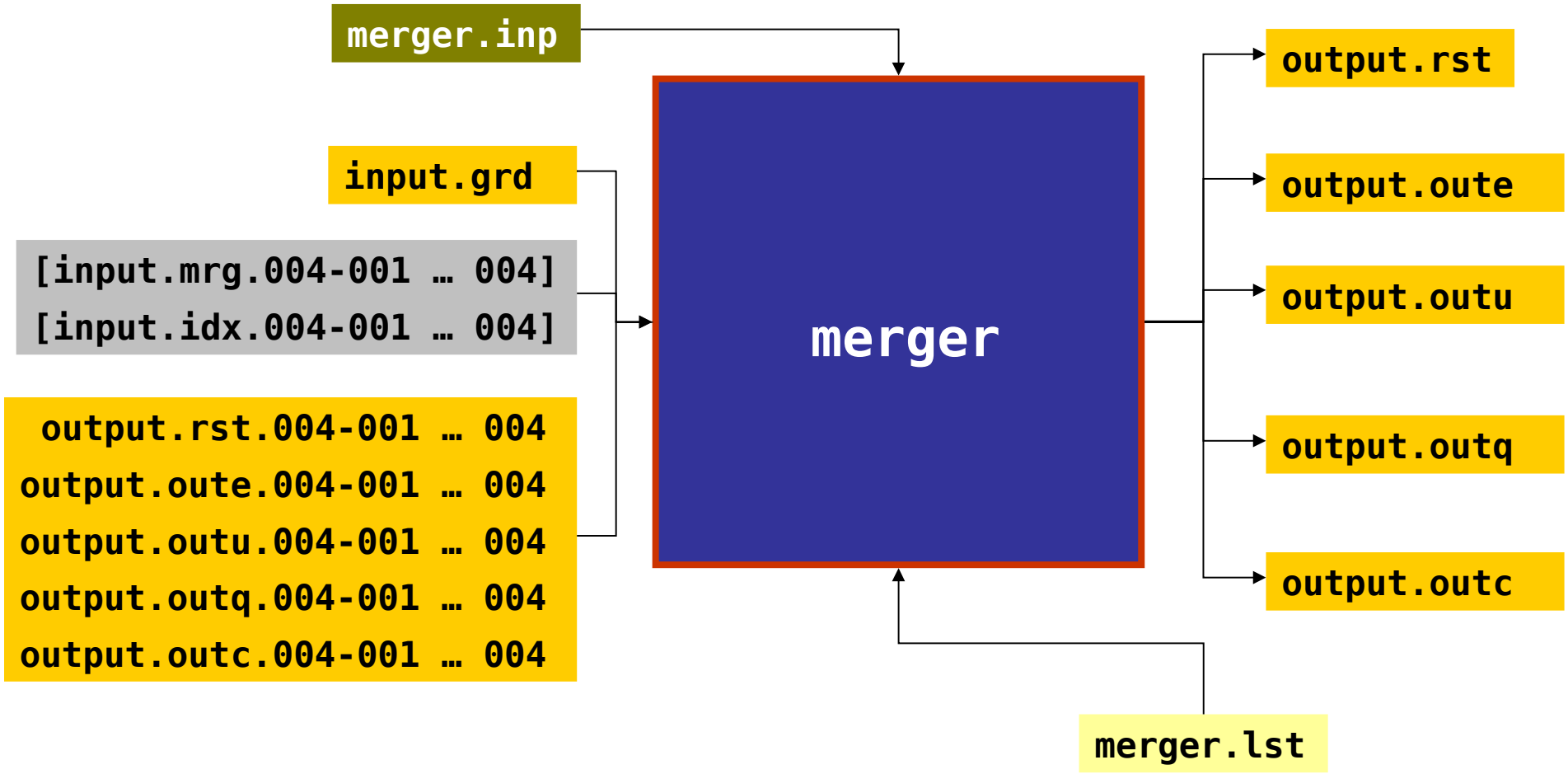
Initial conditions partitioning



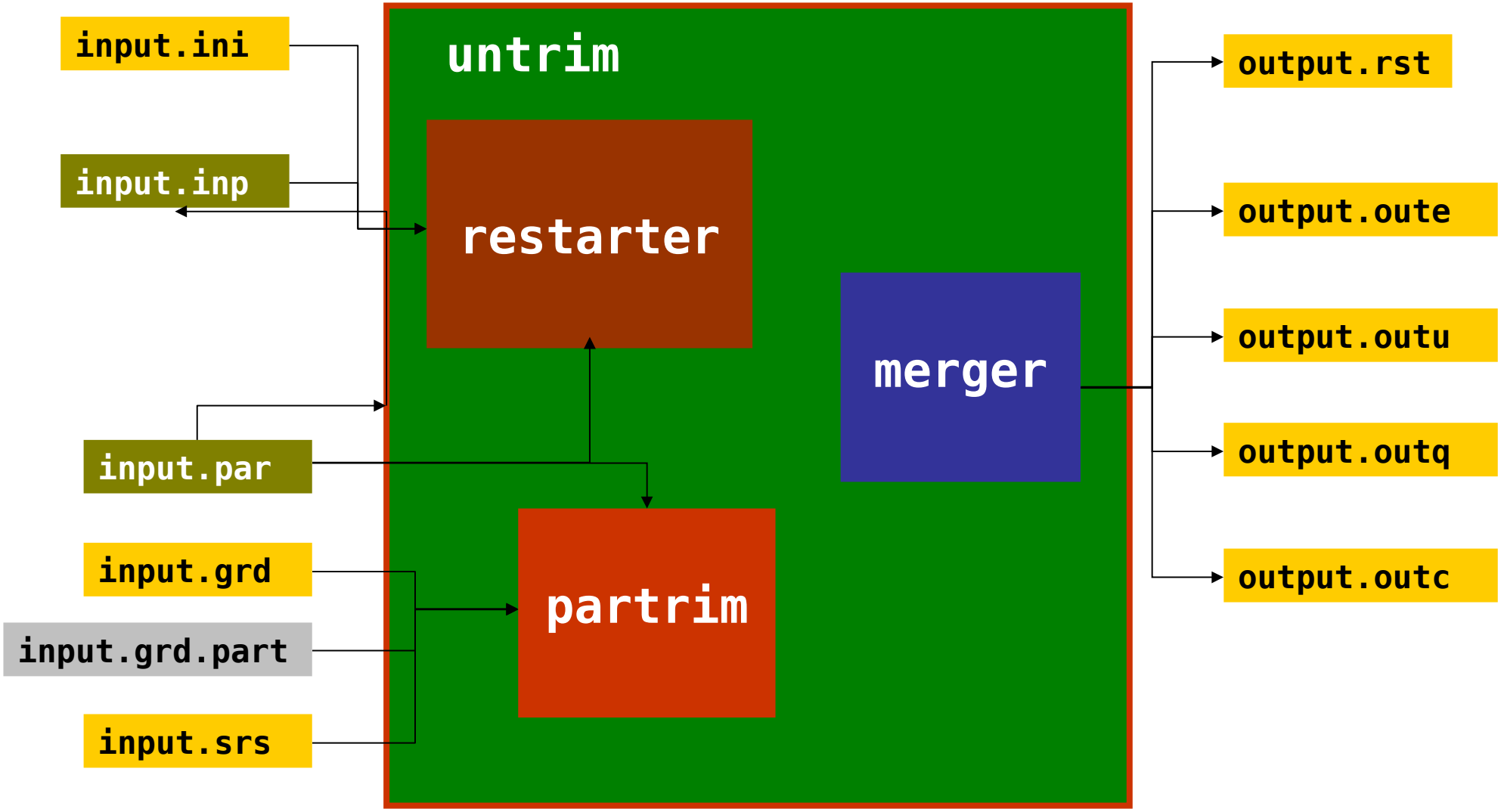
Traditional parallel run – I/O from/to separate files



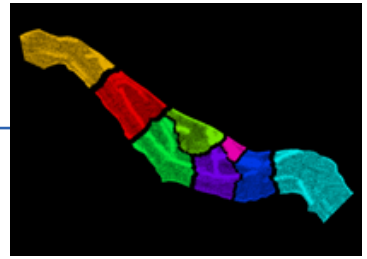
Traditional result merging

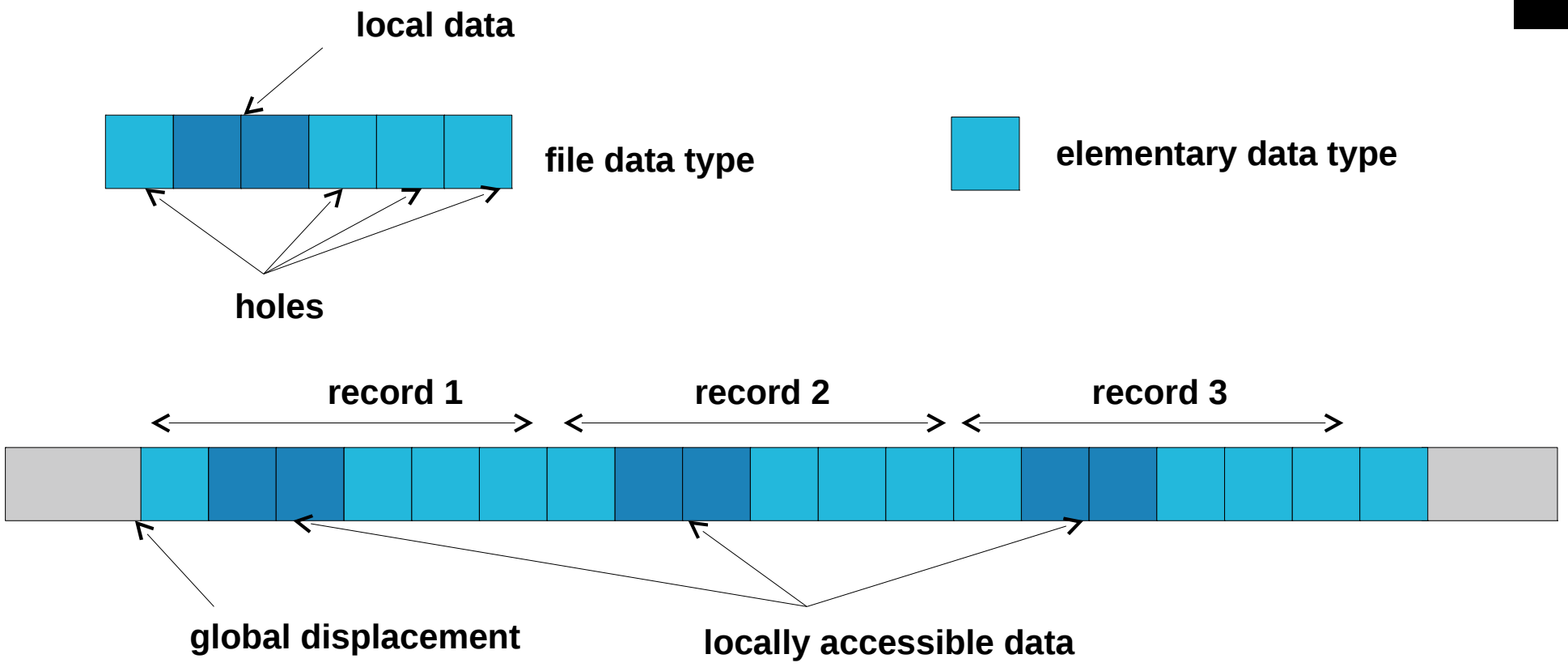


„Automatic“ execution mode



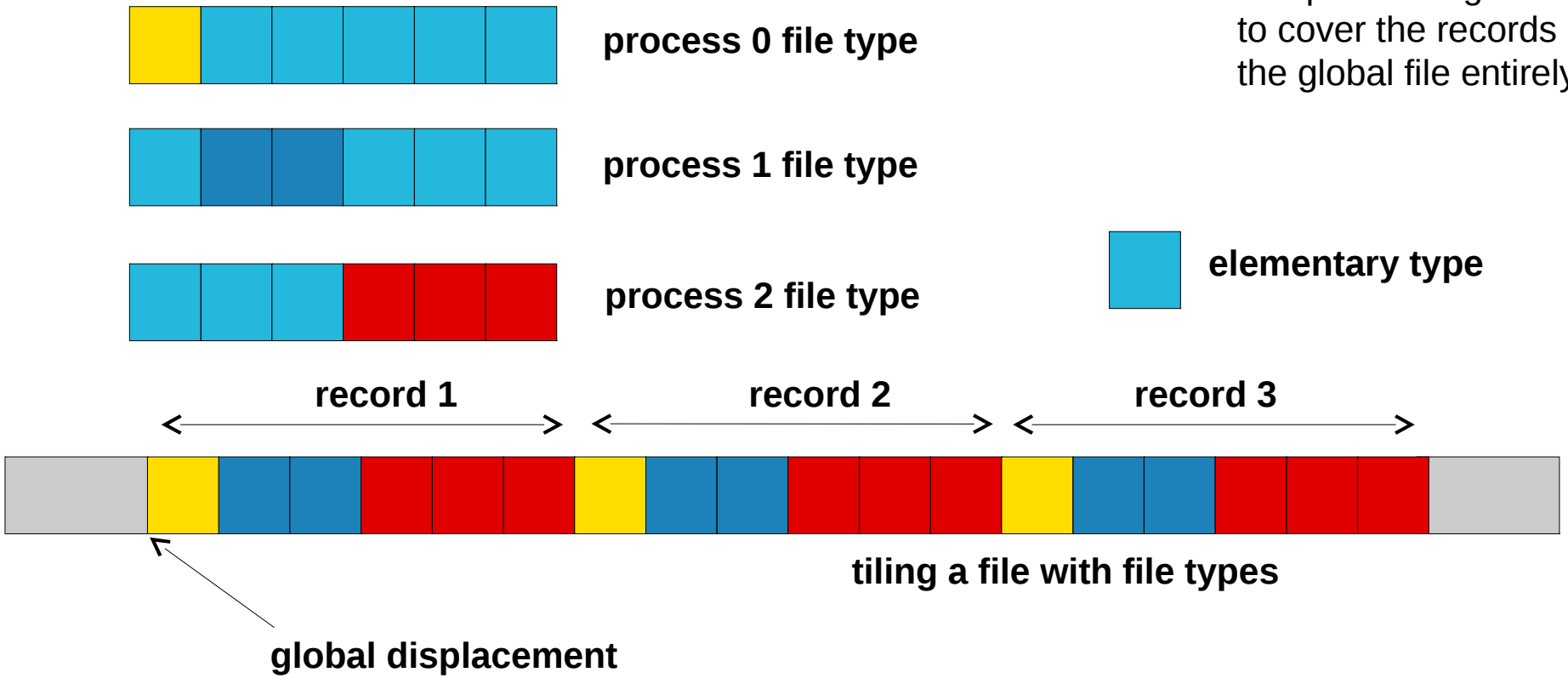
Single rank file view



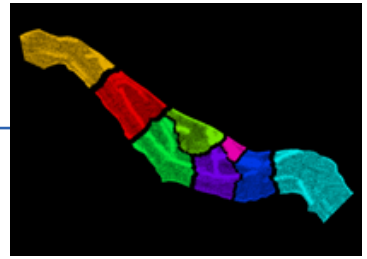


Complementary file types

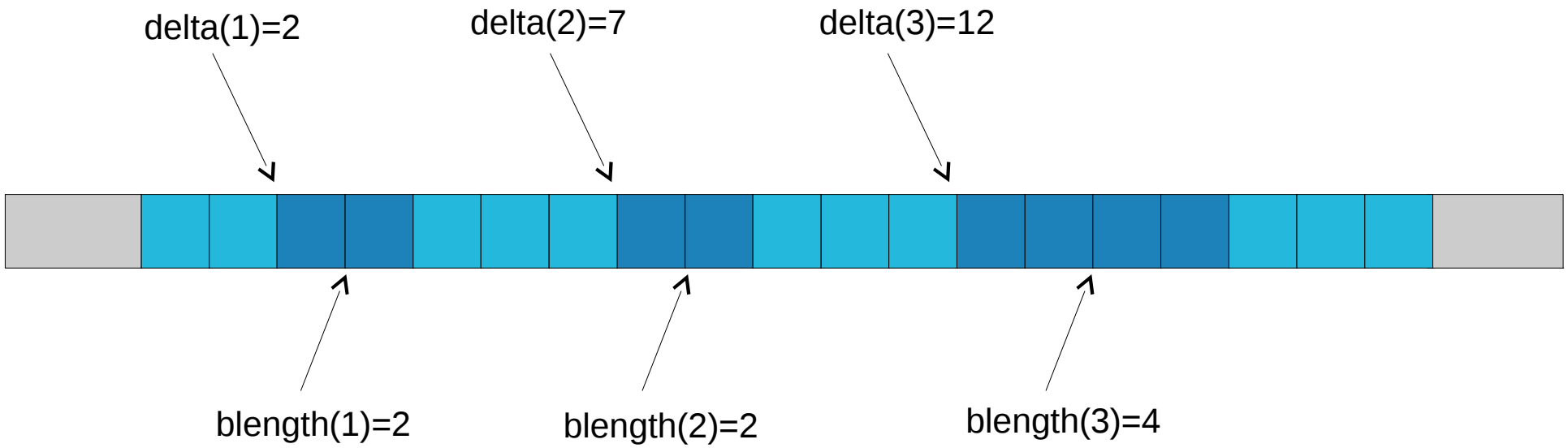
File types of three ranks complementing each other to cover the records of the global file entirely



Committing a file view



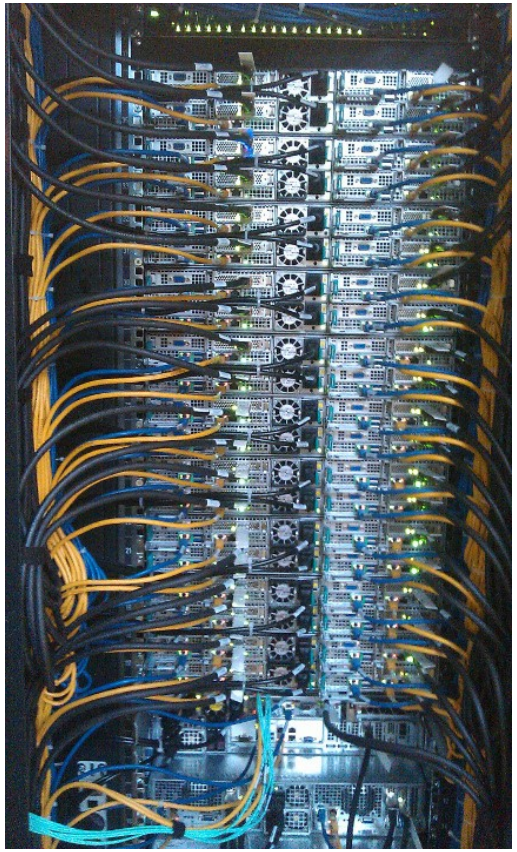
A non-decreasing displacement-blocklength description of the length len=3




```
CALL MPI_Type_Indexed &
   & (len,blength,delta,MPI_REAL8,view,ier)
CALL MPI_Type_Commit(view,ier)
```

Writing a MPI-I/O file

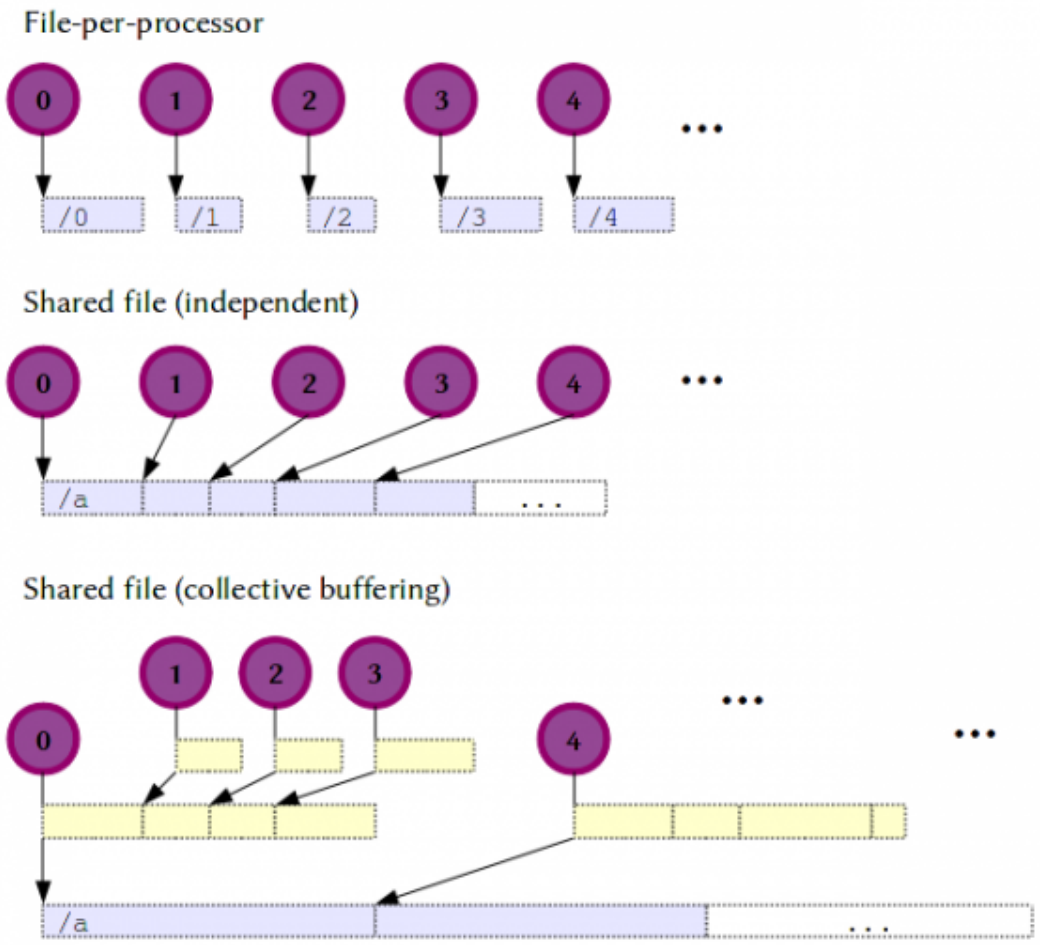
```
CALL MPI_File_Open &
       & (MPI_COMM_WORLD,TRIM(mpiio_restart_file), &
       & IOR(MPI_MODE_CREATE,MPI_MODE_WRONLY), &
       & MPI_INFO_NULL, fhrst, ier)

CALL MPI_File_Set_View &
       & (fhrst, idisp, MPI_REAL8, view, 'native', &
       & MPI_INFO_NULL, ier)

CALL MPI_File_Write_All &
       & (fhrst, iobuffer, lenbuf, MPI_REAL8, &
       & MPI_STATUS_IGNORE, ier)
```



HDF5

- Data model, library and data format for storing and managing scientific data – binary independent cross platforms
- HPC – designed for reading/writing HUGE amount of data
- Data access – collective, independent
- Parallel I/O based on MPI I/O, supports parallel hardware – („hints“, „chunking“)
- Basic metadata description

- Main characteristics https://www.hdfgroup.org/



http://www.nersc.gov/users/training/online-tutorials/introduction-to-scientific-i-o

netCDF: geophysical data

- **NetCDF4** https://www.unidata.ucar.edu/software/netcdf/
  - Network Common Data Form – Unidata 1989-2008 („mature“)
  - large versatility (measurements and output of models as well)
  - Conventions: published agreements how the data should be represented to foster interoperability (e.g. „CF“)
  - commitment to backward compatibility
  - https://www.unidata.ucar.edu/software/netcdf/software.html

- Data containers: ASCII (or native binary data fields) as „legacy“
  - HPC-oriented parNetCDF given up – in favour of HDF5
  - **uses HDF5 as binary data containers with redundant metadata**

Developments presented 2010 still relevant

- *The non-decreasing displacement-length descriptions for local file views can be prohibitively long by arbitrary mesh partitioning.*

- *Countermeasures:*
  - Resorting the original mesh in the sequence of partitions
  - Very short descriptions result (big interior + BCs + interfaces)
  - 2x-8x gain on a non-parallel file system, the larger files the better

- *Some additional aspects irrelevant:*
  - Resorted meshes correct UnTRIM meshes – or not?
  - Maintaining memory-optimised internal sorting (yes!)
  - Overlapping between partitions (halos, interfaces)

A few remarks on I/O

- *Choice of a widely accepted platform-independent format adequate for pedantic inclusion of meta-data, data exchange, archiving: (netCDF)*

- *Choice of the technology allowing speedy read & write on all computer platforms – especially adequate for parallel ones: (HDF5)*

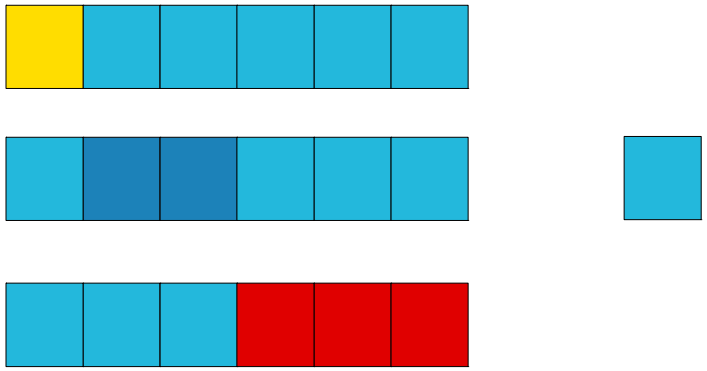
- *For fine-tuned optimisation you still have to:*
  - consider MPI I/O internal requirements
  - formulate “hints” for specific hardware requirements

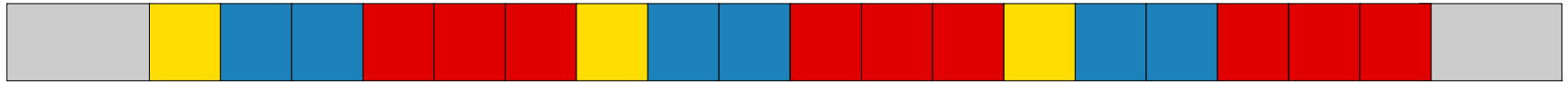
- *Take care of the whole ecosystem around the data and workflows*

If there is still time

Improving and optimising UnTRIM MPI Library, 7th workshop, Trento, 3-5 May **2010**

- **Parallel I/O with MPI**
- (Improving communication itself)





Some outlook

- The *non-mobile* hardware market is driven mainly by the mainstream and consumer *mobile* or *network* computing.

- The cost-aware scientific cluster computing tends to use mostly standard hardware, with specific features or add-ons:
  - speedy internal network *between* the cluster nodes intended to be used in parallel,
  - huge operational memory (distributed, shared),
  - accelerators like standard GPUs but with ECC-memory,
  - multi/manycore processors with longer SIMD units (vector computing),
  - parallel file systems to write/read really huge amount of data.

Some outlook

- The usage of parallelism in algorithms and data is the key to success *(there is no solution for the lack of resolution)*.
- Coarse-grained parallelism is the main method of obtaining speedup *(engineers mostly accept they obtain their results overnight)*.
- UnTRIM is going to stay with us for some more time *(let‘s parallelise it again)*.
- Success after ten years is possible.

Source: Man Booker Prize

A success *ten years after* is possible

- Author Olga Tokarczuk wins with translator Jenniffer Croft Booker Prize **2018**, for a book „Flights“ *[Bieguni]*, published in **2007**.
- Numerous bad, bad critic voices: „Only a few readers will manage to read the book to the end“, etc.
- It received both *readers and jury* parts of Polish highly regarded Nike Prize 2008.
- Tokarczuk asked 2018 about a book published in 2007: ***Sorry, but I am conversationally jet-lagged…***

The evil is at its strongest when you don`t move.
You cannot defend yourself, you have to take flight:
Whoever stops – will turn into stone.
Whoever tries to rest for a while – will be pinned like an insect…

Polish author Olga Tokarczuk, left, stands with translator Jennifer Croft after wiinning the Man Booker Internationnal prize 2018, for her book *Flights*, at Victoria and Albert Museum, May 22, 2018 in London

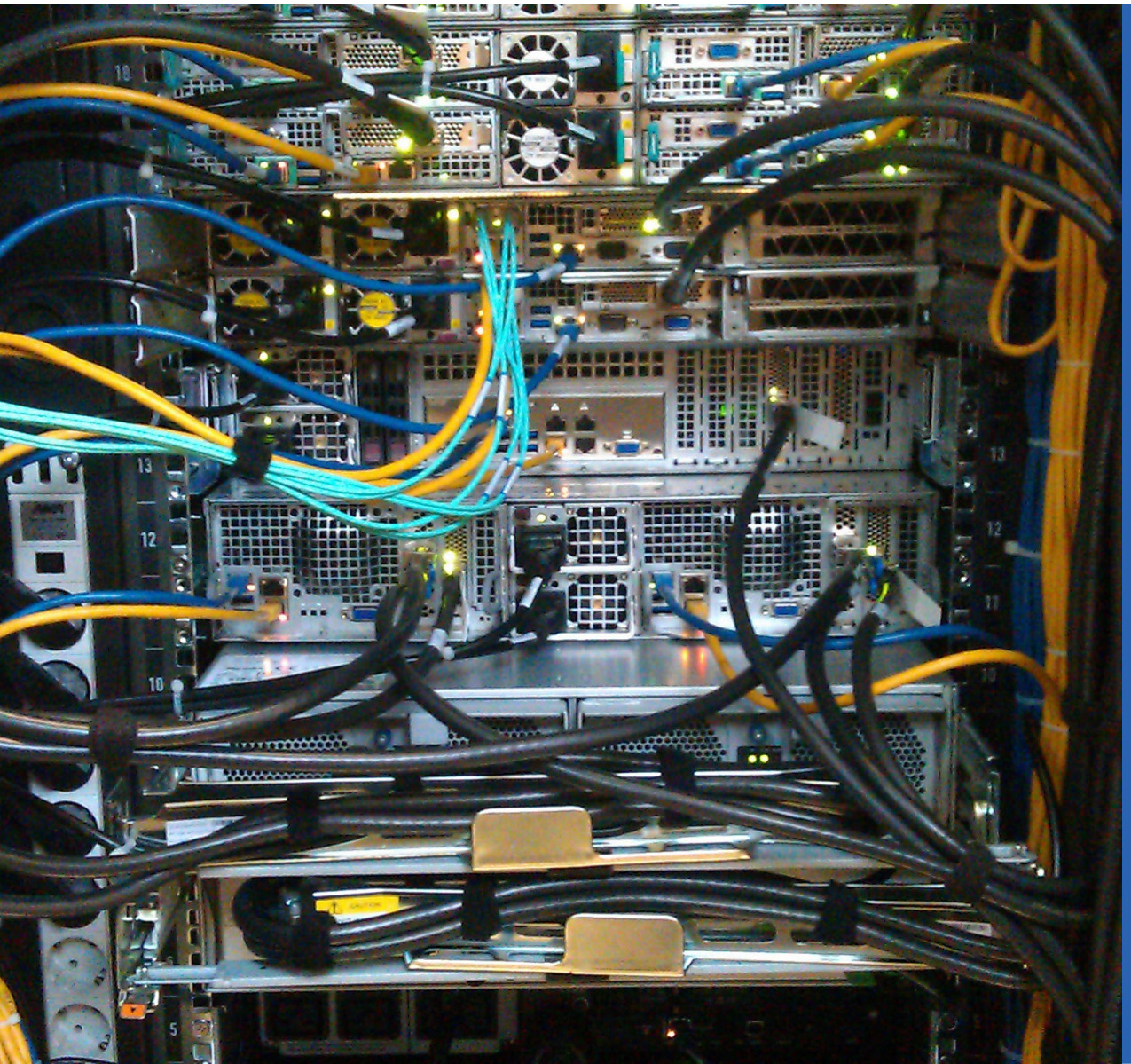
[Source: Janie Airey / Man Boooker Prize

Antoine de Saint-Exupéry knew it better

An administration,
like a machine,
does not create.
It carries on.

[Source: Wikipedia]

French last 50 francs banknote



Ten Years After: MPI UnTRIM's past and future

Bundesanstalt für Wasserbau
76187 Karlsruhe, Germany

www.baw.de