# How to overcome common performance problems in legacy climate models

Jörg Behrens[1]

Contributing:

Moritz Handke[1], Ha Ho[2], Thomas Jahns[1], Mathias Pütz[3]

1 Deutsches Klimarechenzentrum (DKRZ)

2 Helmholtz-Zentrum Geesthacht (HZG)

3 CRAY, formerly IBM

# How to overcome common performance problems in legacy climate models

## Problems:

- *Legacy* models: adapted to past hardware generations
- Today: much more cores, models do not scale
- #{models}  x  #{performance-problems} > manpower
  - Solution path: improve **software infrastructure** (libraries)
- Order and number of bottlenecks change with #{cores}
  - Performance **analysis** comes first

# Outline

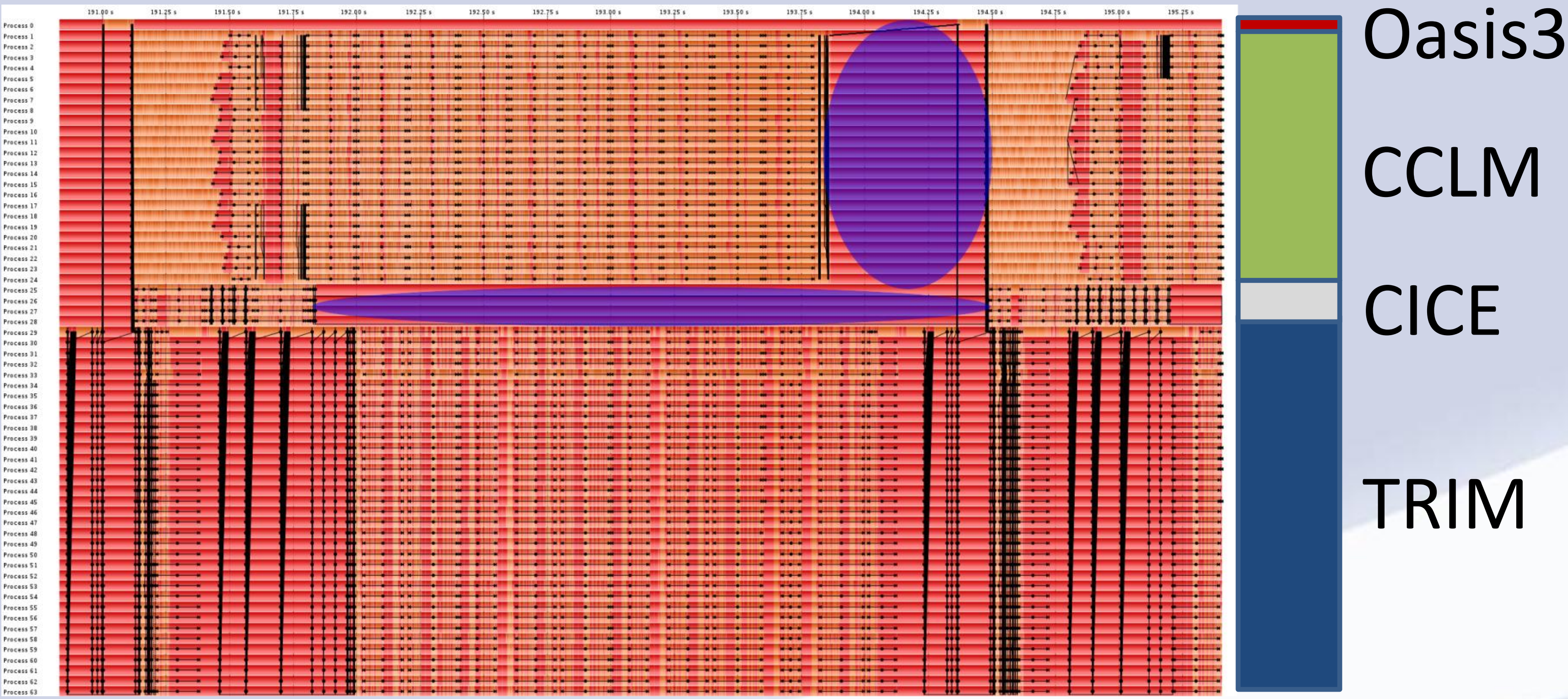
- Quick performance analysis:
  - Example (TRIM: "Tidal, Residual, and Intertidal Mudflat Model“ )
- Problems & solutions
  - ➢ More analysis (dynamic load imbalance)
- Performance software infrastructure
  - Motivating example (MPIOM)
  - ➢ YAXT (Yet Another Exchange Tool)
- Quick view on indirect data access

# Performance analysis example

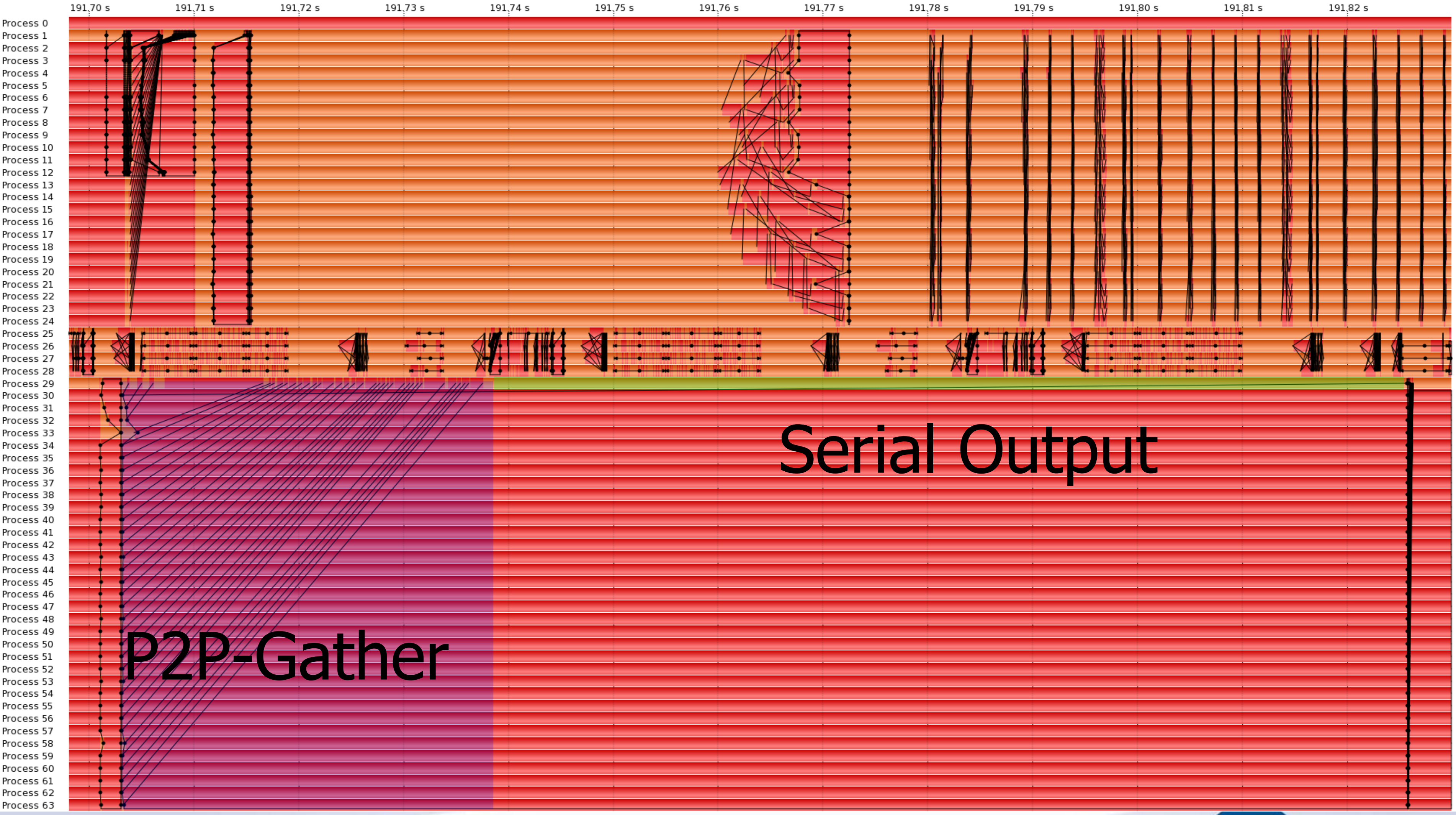
## **From:** Program Analysis and Tuning Workshop (DKRZ, June 2012)

User were invited to analyze their applications using Vampir / Scalasca.

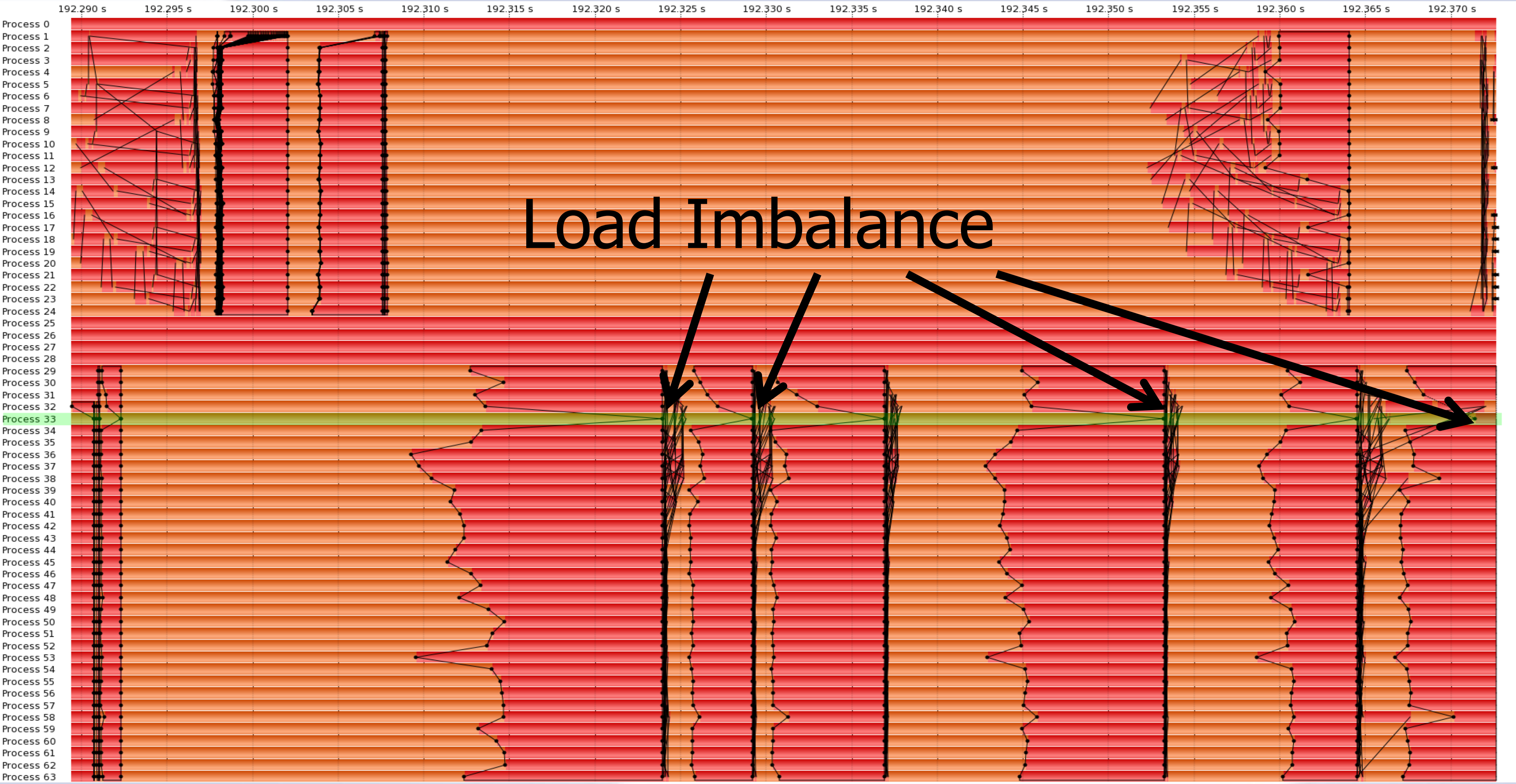
Example: coupled model (TRIM & CICE & CCLM & OASIS3)



# Performance problem pattern (1)



# Performance problem pattern (2)

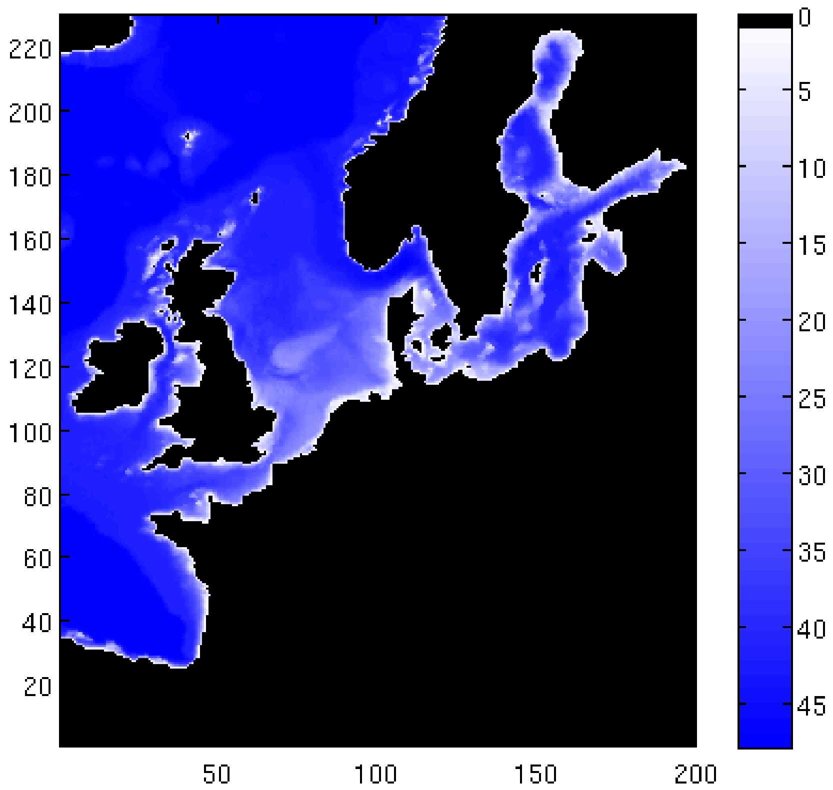


# Solutions to problems 1 & 2

1. p2p-gather & serial IO:
   - Quick: use ScalES* 2-phase gather (much faster)
   - Better: parallel IO (not discussed here)
2. Load imbalance:
   - Analyze further: Decomposition? Cost-function?
   - Inspect source code
   - Detailed measurement of work load

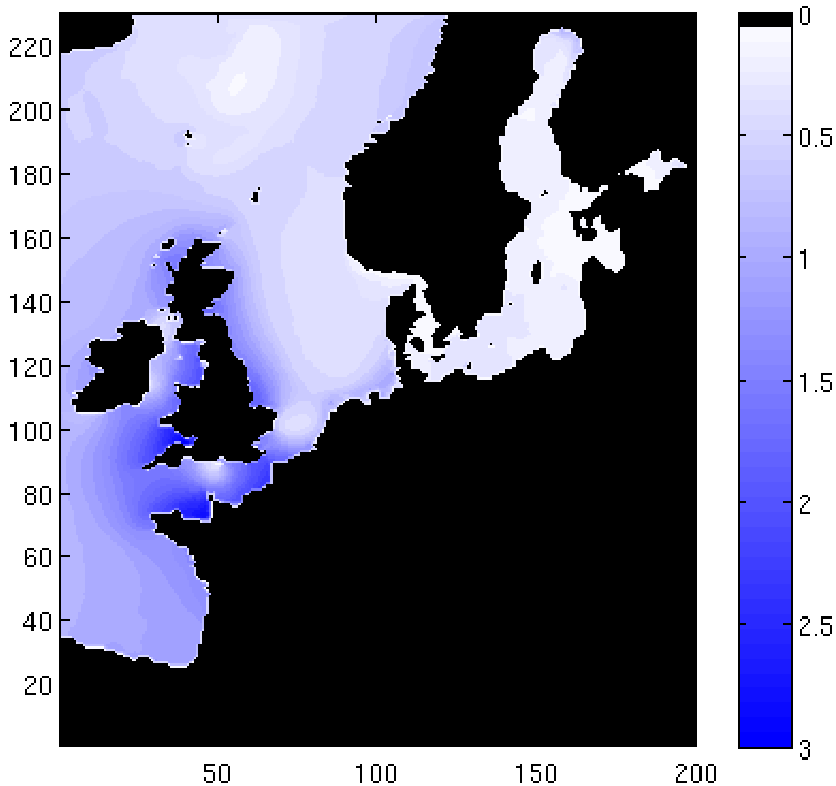
*Scalable Earth System Models

# TRIM: average work load



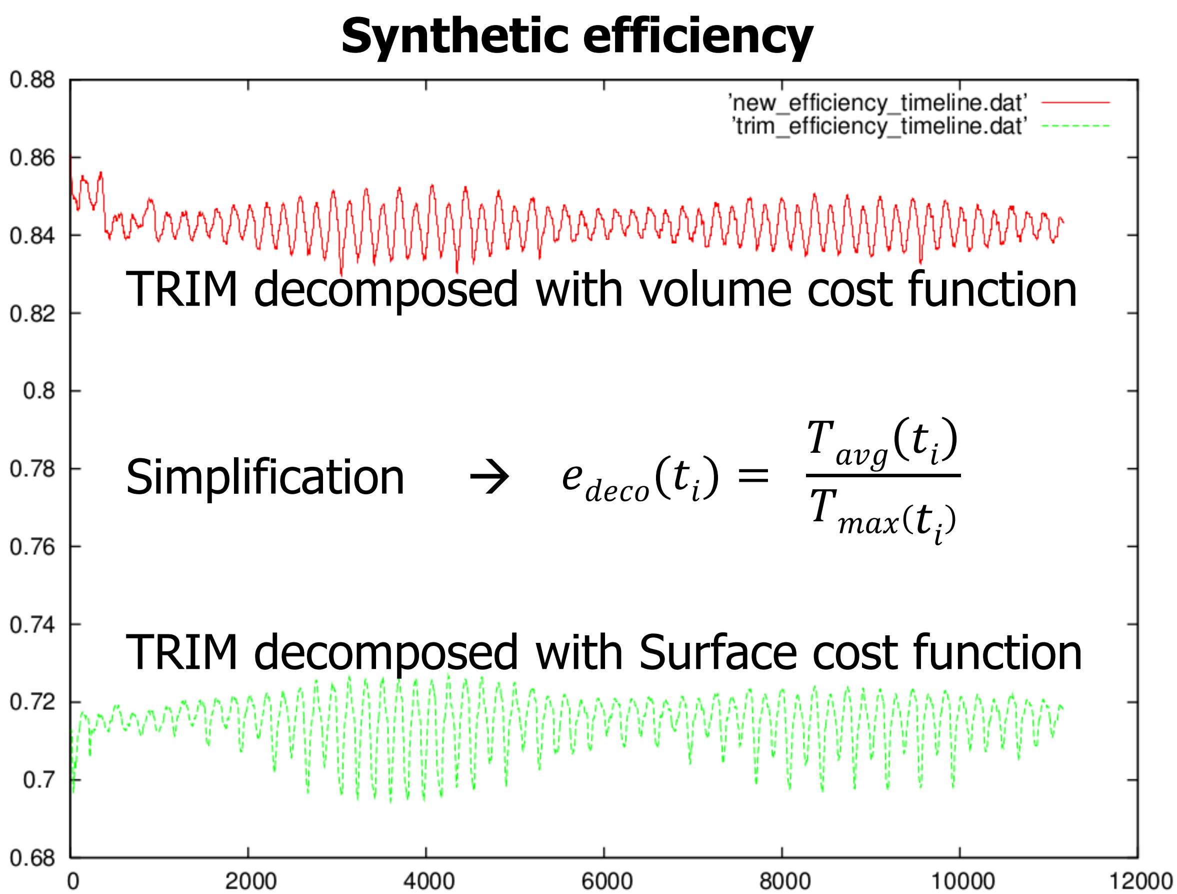

- $nlev(t)_{i,j} \sim cost(t)_{i,j}$
- Model: (0/1 land-sea mask) cost function for partitioner
- Partitioner gives suboptimal decompositions
- ➢ No Problem for low nprocs
- ➢ But steals efficiency

# TRIM: σ[work load]




- $\sigma_{i,j}$ over one month
  - Low nlev-variation
  - Error of a static decomposition should be small enough to justify work on improvement.
  - (Dynamic load-balancing would require major rewrite)

# TRIM: iteration count → efficiency



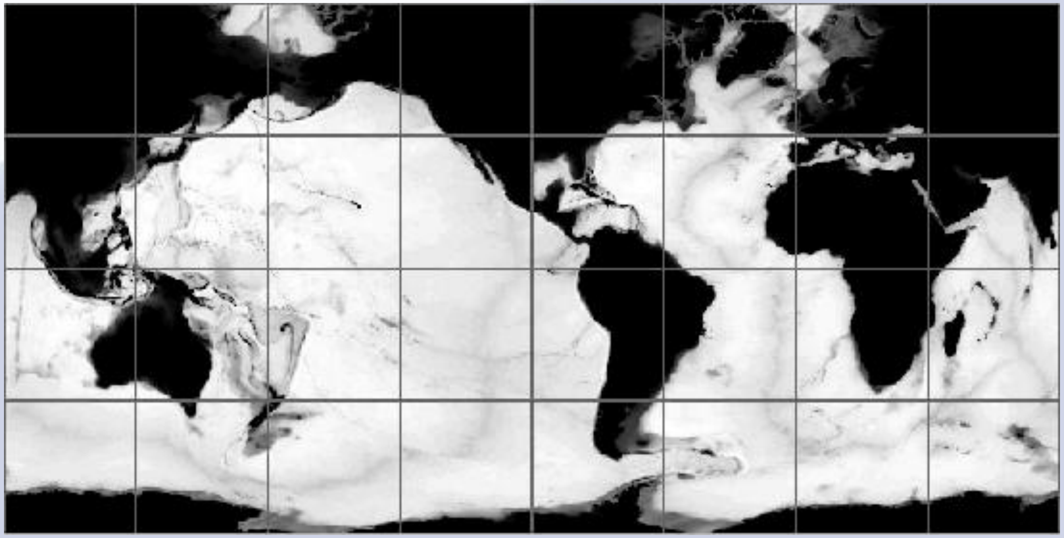

Simplification → $e_{deco}(t_i) = \frac{T_{avg}(t_i)}{T_{max(t_i)}}$

**Can be improved to 91% by using the next magic partition numbers (16x8)**

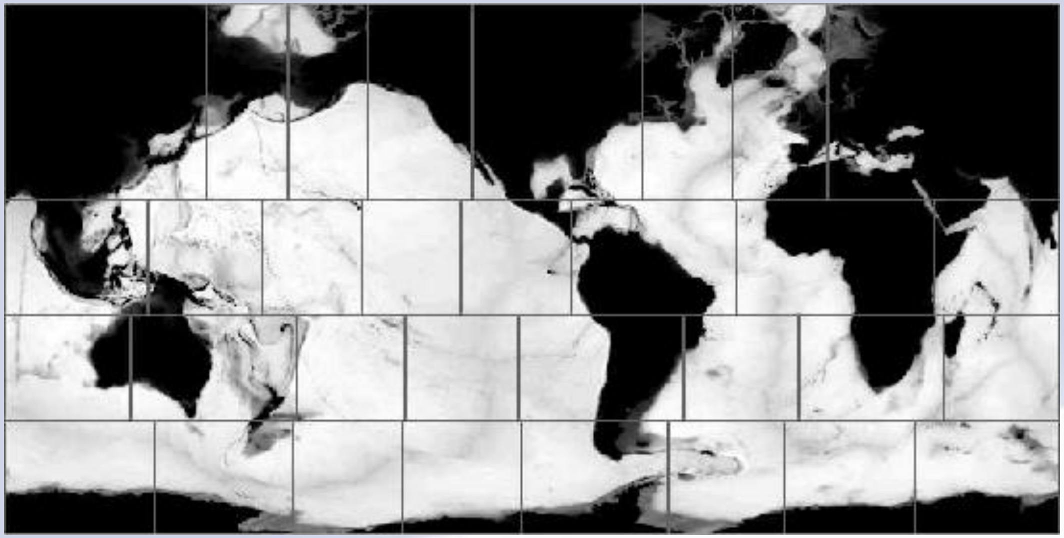


DKRZ

# MPIOM [TP04L40: 8x4] Load Balance



1. Wet-point-only optimized
   - workload changed
   - Unfit legacy decomposition
   - Nothing gained



2. Adapted decomposition
   - Old boundary exchange fails
   - Reprogramming exchange?
   - YAXT-formulation:
     - works for both cases
     - faster

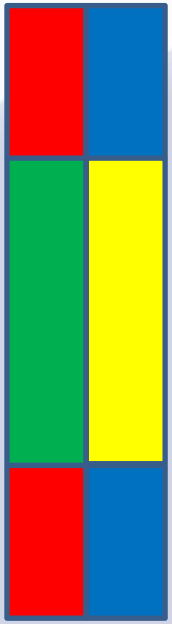
# YAXT (Yet Another Exchange Tool)

- Successor of UniTrans (ScalES prototype)
- General communication library on top of MPI
  - **C** with Fortran interfaces
- Simple definition of communication
  - Change: programming communication → parameterization of comm.
- Fast & flexible communication made easy
  - Automatically generates derived MPI datatypes
  - Easy aggregation of communication steps

# UniTrans/YAXT: key concepts

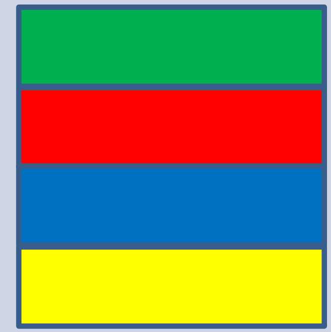
- Global domain: **indexed elements** $\{e_i\}$
- Source and target indices: $\{s_i\}$, $\{t_i\}$ per process
- Automatic construction of **index map**
  ~ Extended communication matrix
- Automatic or user-supplied (static) data offsets:
  - local index position → local data position
  - *index map* → **data redistributor**
- Automatic generation of optimized MPI-datatypes
- **Aggregation** of several data redistributors
- Focus on **static communication** pattern
  – Initialization is considered a one time cost

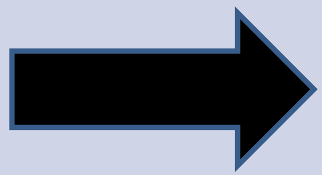
# YAXT – communication objects



Source decomposition

Target decomposition





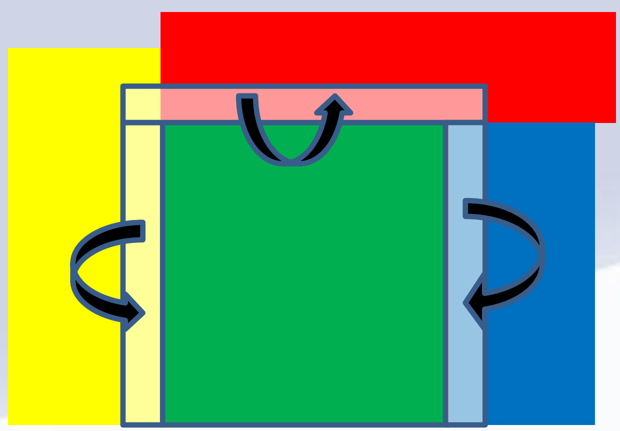
Transposition: Data redistributor

Source decomposition:

- Core points

Target decomposition:

- Halo points



Hallo update

# Complexity of data exchange

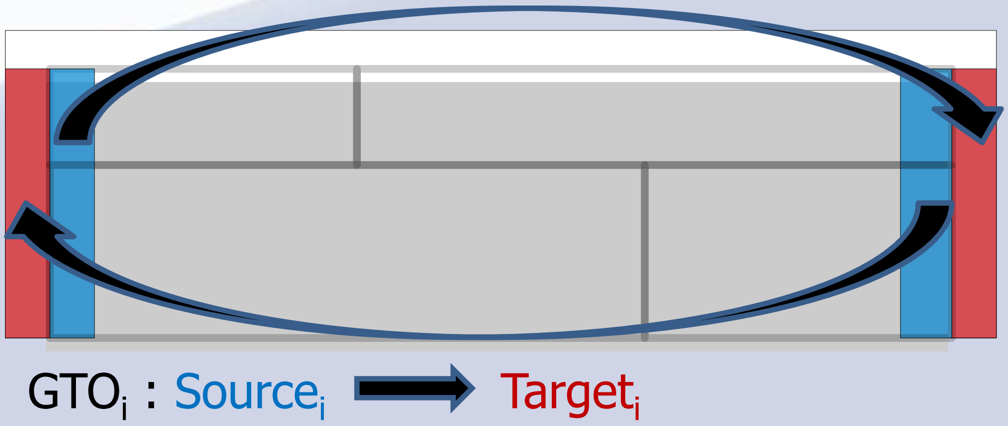
Some aspects:

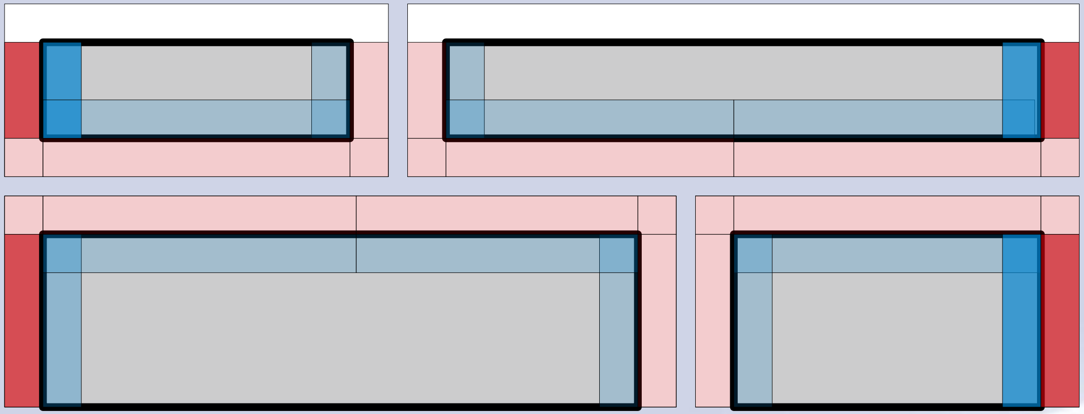
- Topology & physical quantities
  - not always simple (e.g.: tripolar grid)
  - possible sign change at boundary
- Stencil shape
  - depends on physical operator
  - ➢ halo update requirements
- Subdomain boundaries
  - given by partitioner
  - ➢ Communication matrix

Independent of parallelism (serial concept)

depends on parallelism (parallel concept)

# Simplified generation of communication objects (prototype)



**Input:**

- Global topology (GTOs)
- Stencil shape
- Local subdomain boundaries

**Output:**

- Data redistributor for halo update

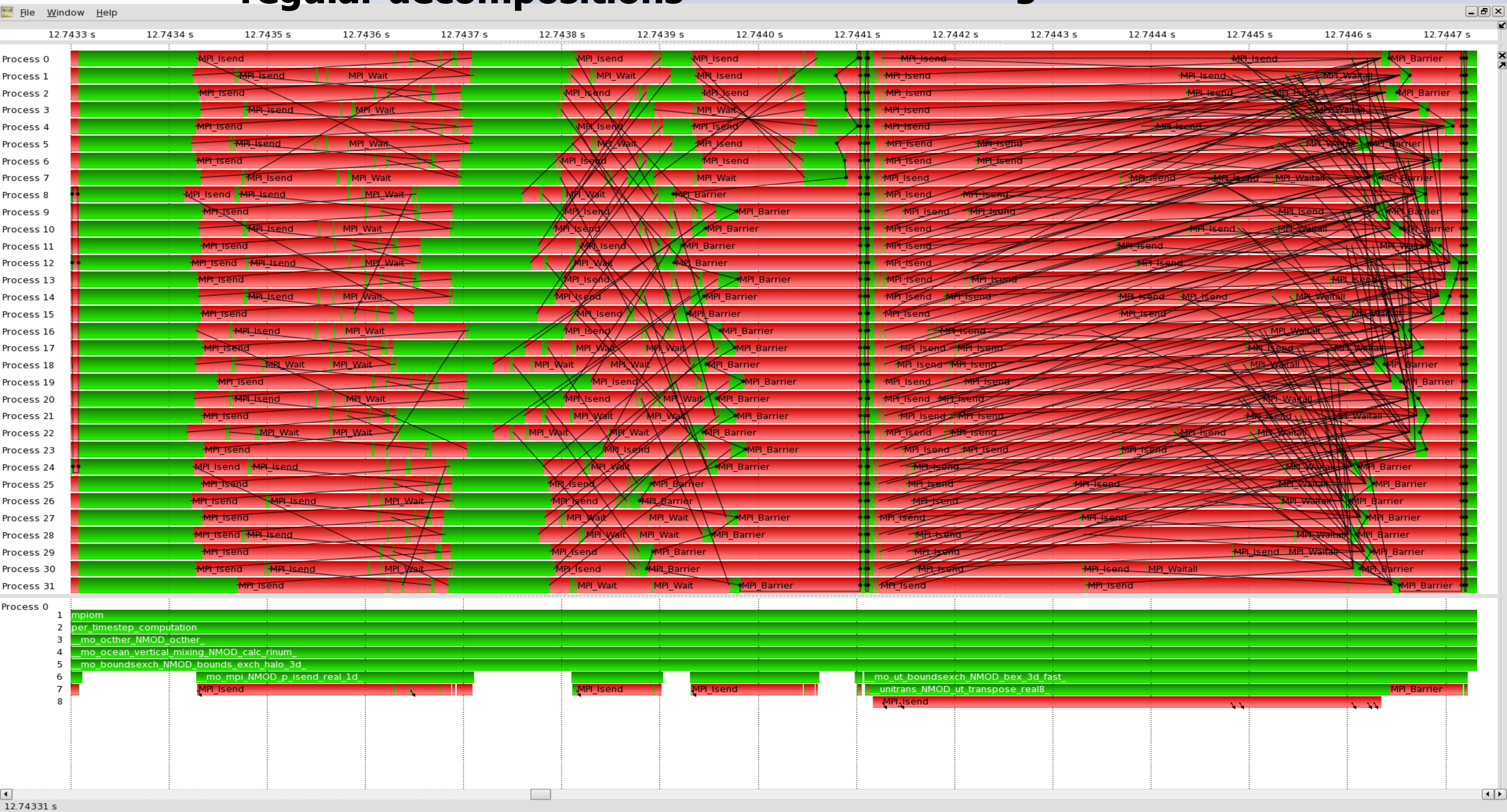
**possible data representations:**

- Simple arrays, or
- derived element offsets

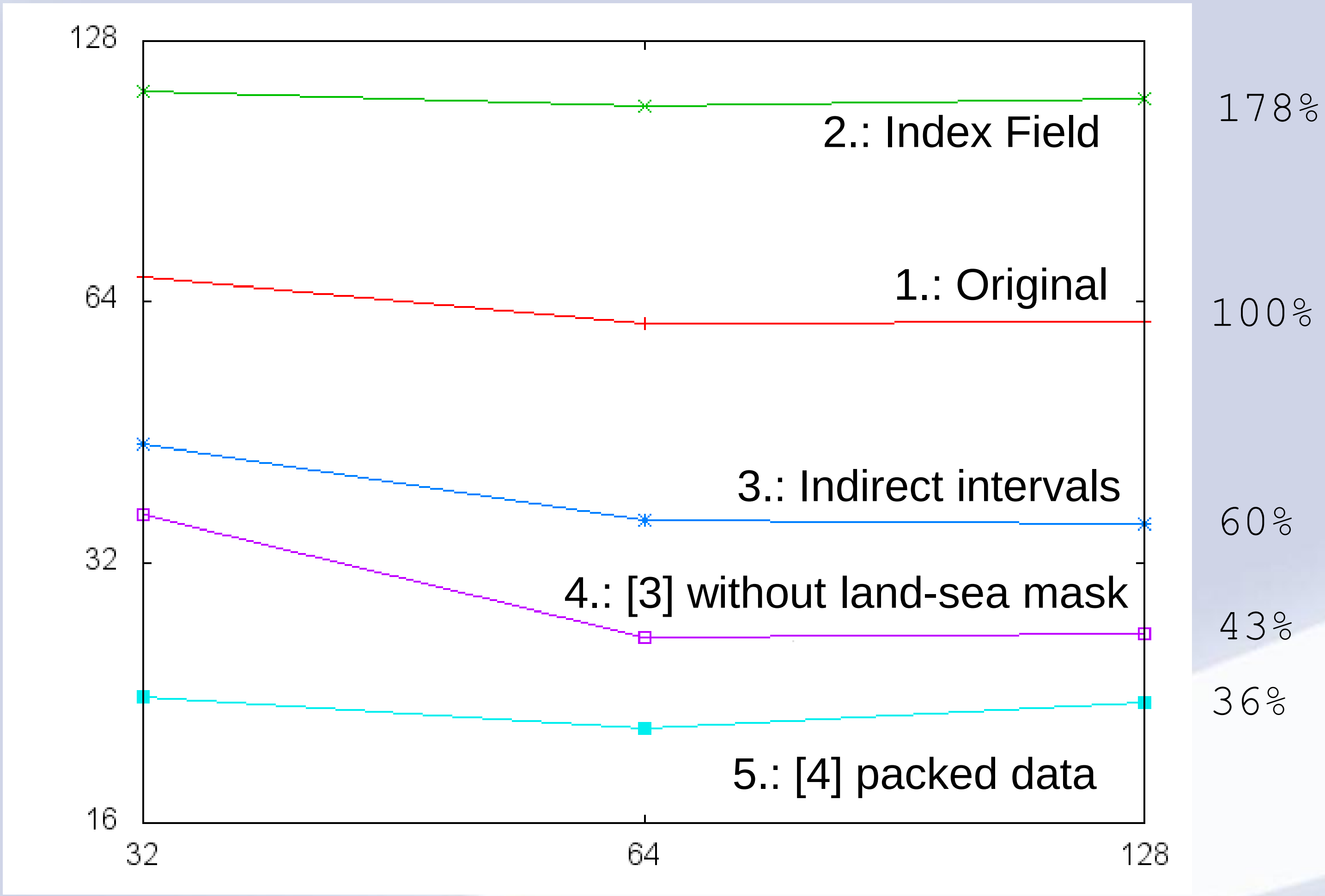
# Example: MPIOM boundsexchnage

**Handwritten – requires symmetric regular decompositions**

**UniTrans/YAXT general solution**



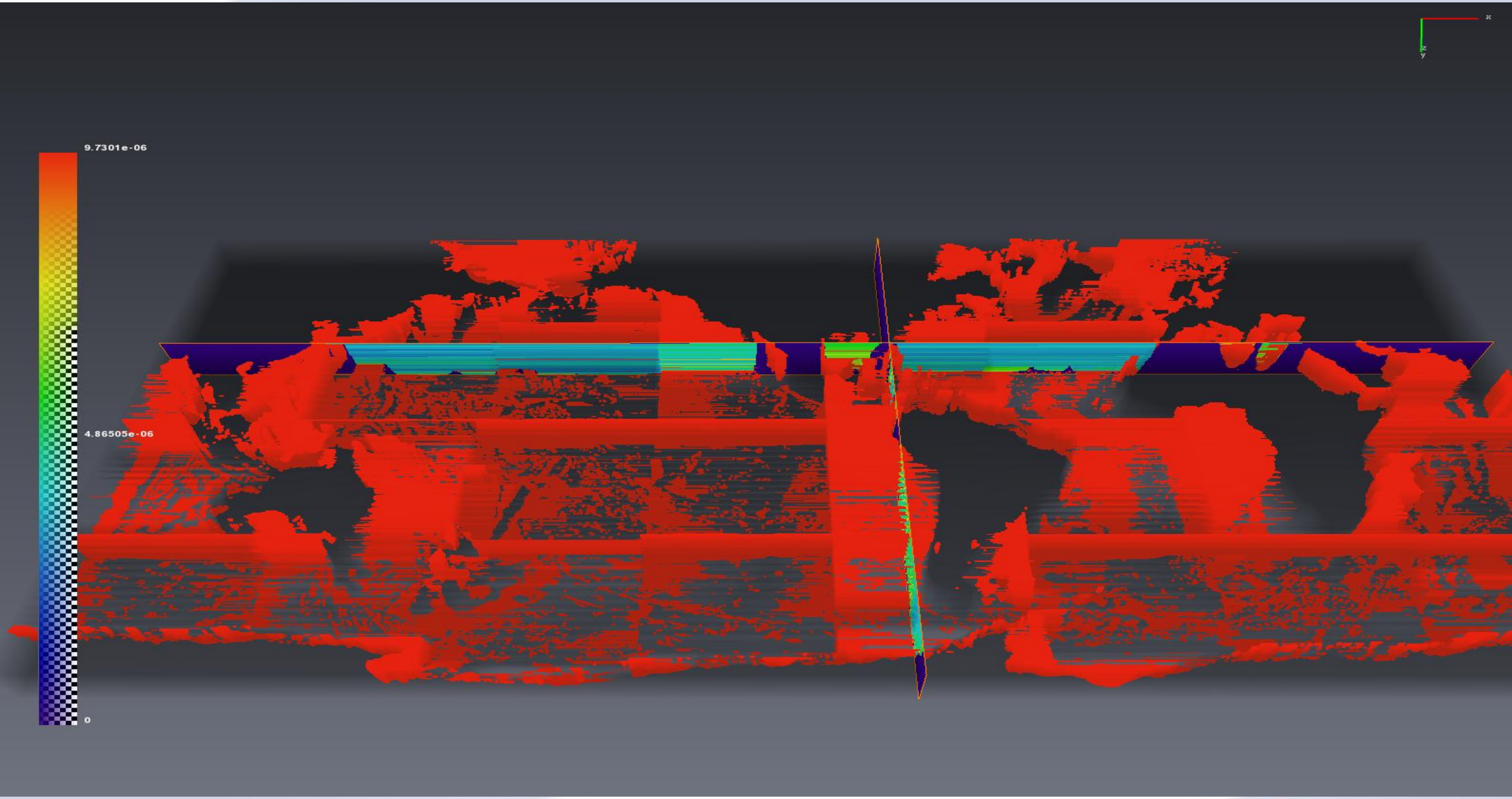
# Indirect data access
# MPIOM: baroclinic kernel, example






DKRZ

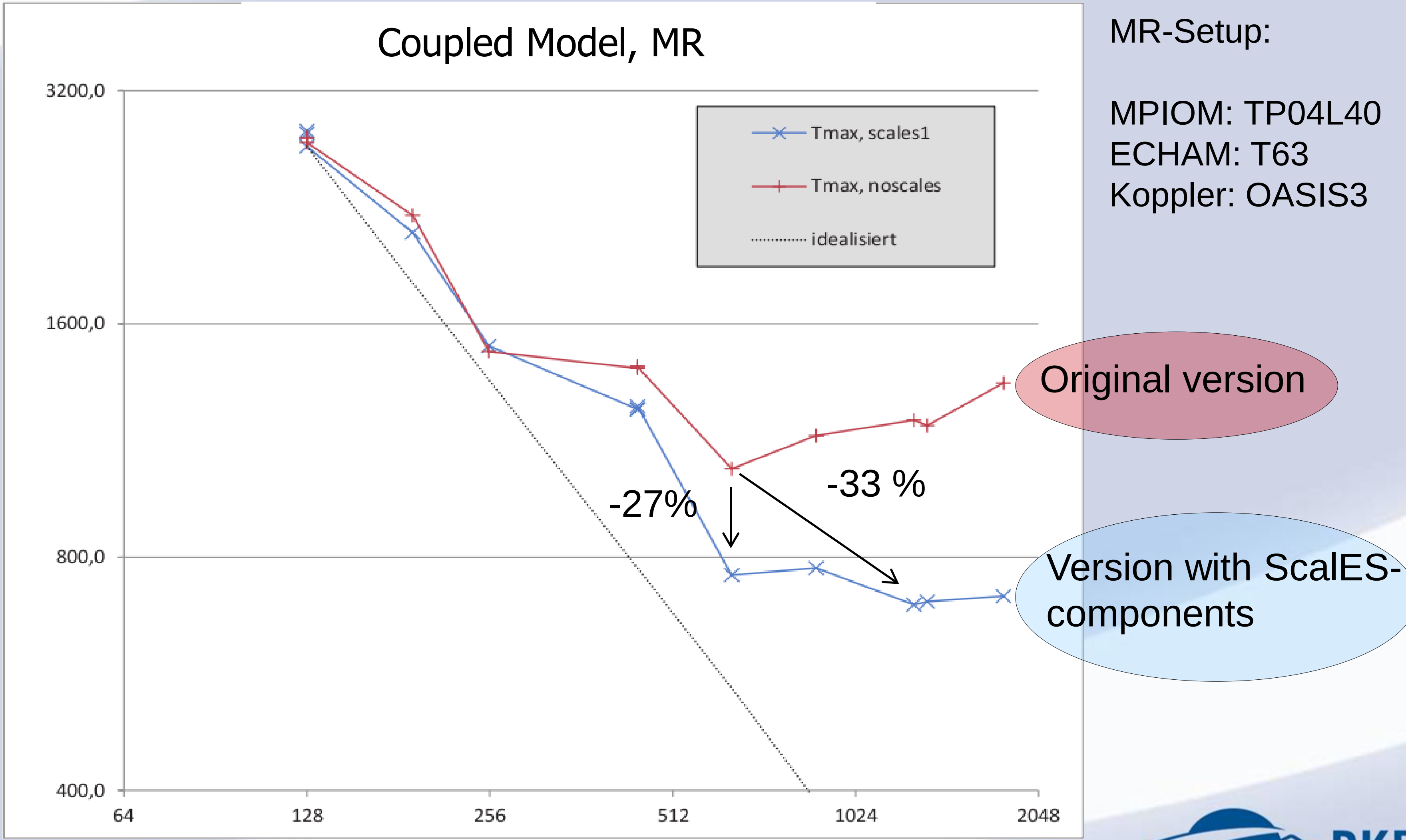
# Performance Visualization
## (MPIOM: selected 3d-loop: iteration cost)






DKRZ

# COSMOS runtime with improved communication




MR-Setup:

MPIOM: TP04L40
ECHAM: T63
Koppler: OASIS3





DKRZ