Scalable Earth-System-Models for high productivity climate simulations

Panagiotis Adamidis[2] Dirk Barbi[1] Jörg Behrens[2] Joachim Biercamp[2] Kerstin Fieg[1,2] Thomas Jahns[2] Vincent Heuveline[6] Wolfgang Hiller[1] Christoph Junghans[3] Deike Kleberg[5] Luis Kornblueh[5] Hendrik Merx[4] Mathias Puetz[3] Benedikt Steil[4] Florian Wilhelm[7]
Alfred-Wegener-Institut für Polar- und Meeresforschung, Bremerhaven[1]; Deutsches Klimarechenzentrum, Hamburg[2]; formerly IBM Deutschland GmbH, Ehningen[3]; Max-Planck-Institut für Chemie, Mainz[4]; Max-Planck-Institut für Meteorologie, Hamburg[5]; Karlsruher Institut für Technologie, Karlsruhe[6], formerly[7]

Introduction

The purpose of the ScalES project was to create reusable software components to solve recurring issues in parallelizing and optimizing climate models. The project was proposed to address the following topics in particular:

- Parallel I/O
- Load Balancing
- Coupling
- Optimizations for massively parallel systems (exploit advanced communication patterns and memory hierarchy)
- Demonstrate viability of the solutions in the widely used MPI-ESM (then named COSMOS)

To address these issues, the following software components were created or extended:

- **CDI-pio**, based on CDI (**c**limate **d**ata **i**nterface library) is a parallelized extension of the CDI GRIB and netCDF I/O routines.
- **ScalES-PPM** (**p**artitioning and **p**arallelization **m**odule) is a library consisting of descriptors for various partitioning schemes and a collection of parallel, mixed-mode solvers.
- **YAXT** (**y**et **a**nother e**x**change **t**ool) allows for a simplified formulation of efficient MPI communication for data parallel applications. YAXT is an extension and generalization of the original Fortran prototype implementation (Unitrans).
- The **OASIS 4** coupler was adapted for the triangle-based grid of the **F**inite-**E**lement **S**ea-**I**ce **O**cean **M**odel (**FESOM**) to couple FESOM with ECHAM, the atmospheric component of MPI-ESM.
- An advanced Scatter-Gather facility that exploits the inherent regularities of MPI-ESM decompositions.

Parallel Solver in ScalES-PPM

ScalES-PPM includes the following functionality:

- Solvers:
  - Conjugate-Gradient Method
  - Chebyshev Iteration
  - Additive Schwarz Method
  - Multi-Precision Iterative Refinement
- Preconditioners:
  - Jacobi
  - Block Symmetric SOR
  - Block Incomplete LU with fill-in 0
  - Block Incomplete Cholesky with fill-in p
  - Block Modified Incomplete Cholesky with fill-in p
- Evaluation:
  - Functions to calculate λ_{min} and λ_{max} of preconditioned systems.

The solvers have been validated on and performance tested with MPIOM TP6ML20 on DKRZ blizzard. An FPGA accelerator platform was also evaluated for potential performance benefits.

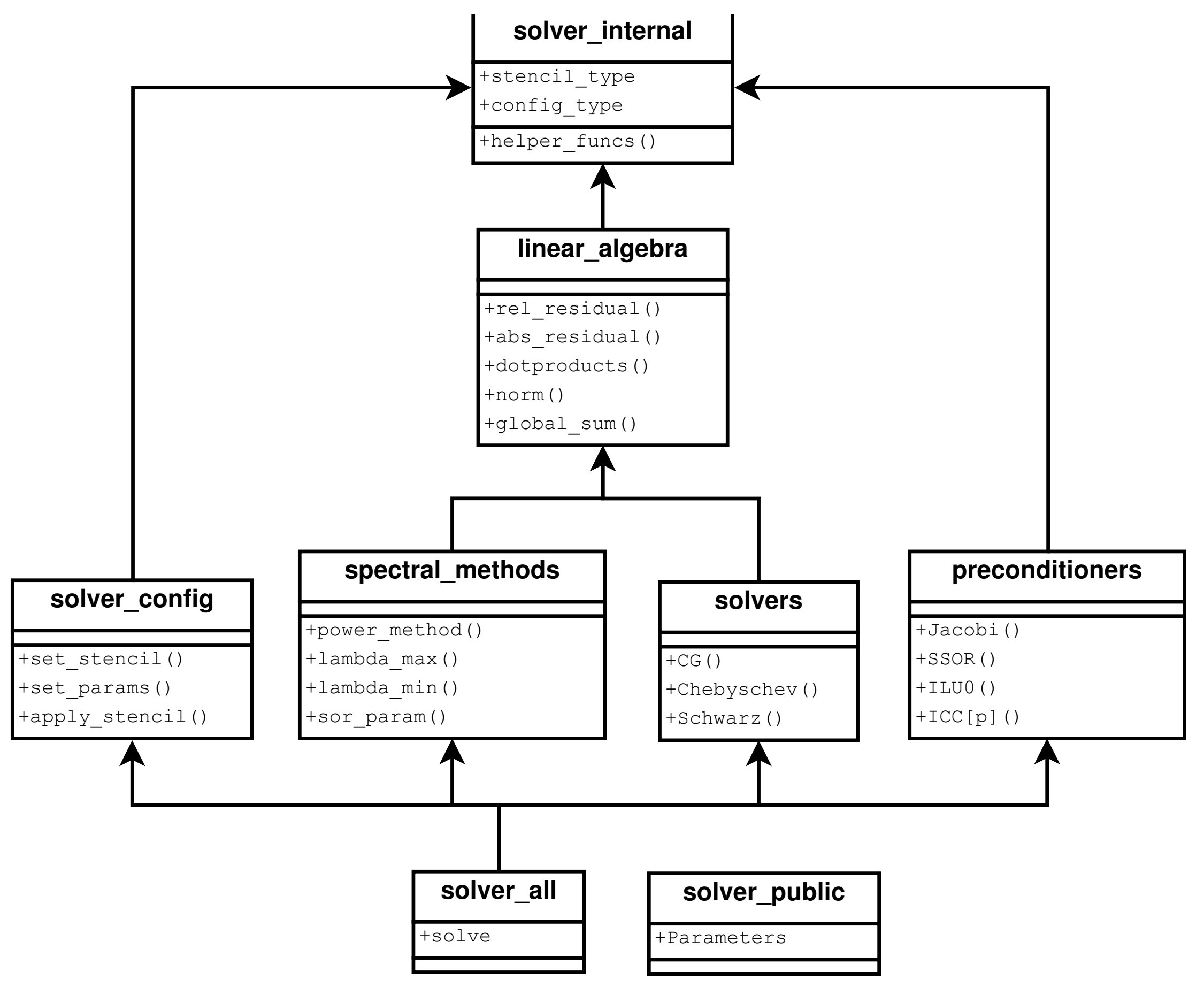

Figure: The solver functions are structured in an extensible manner

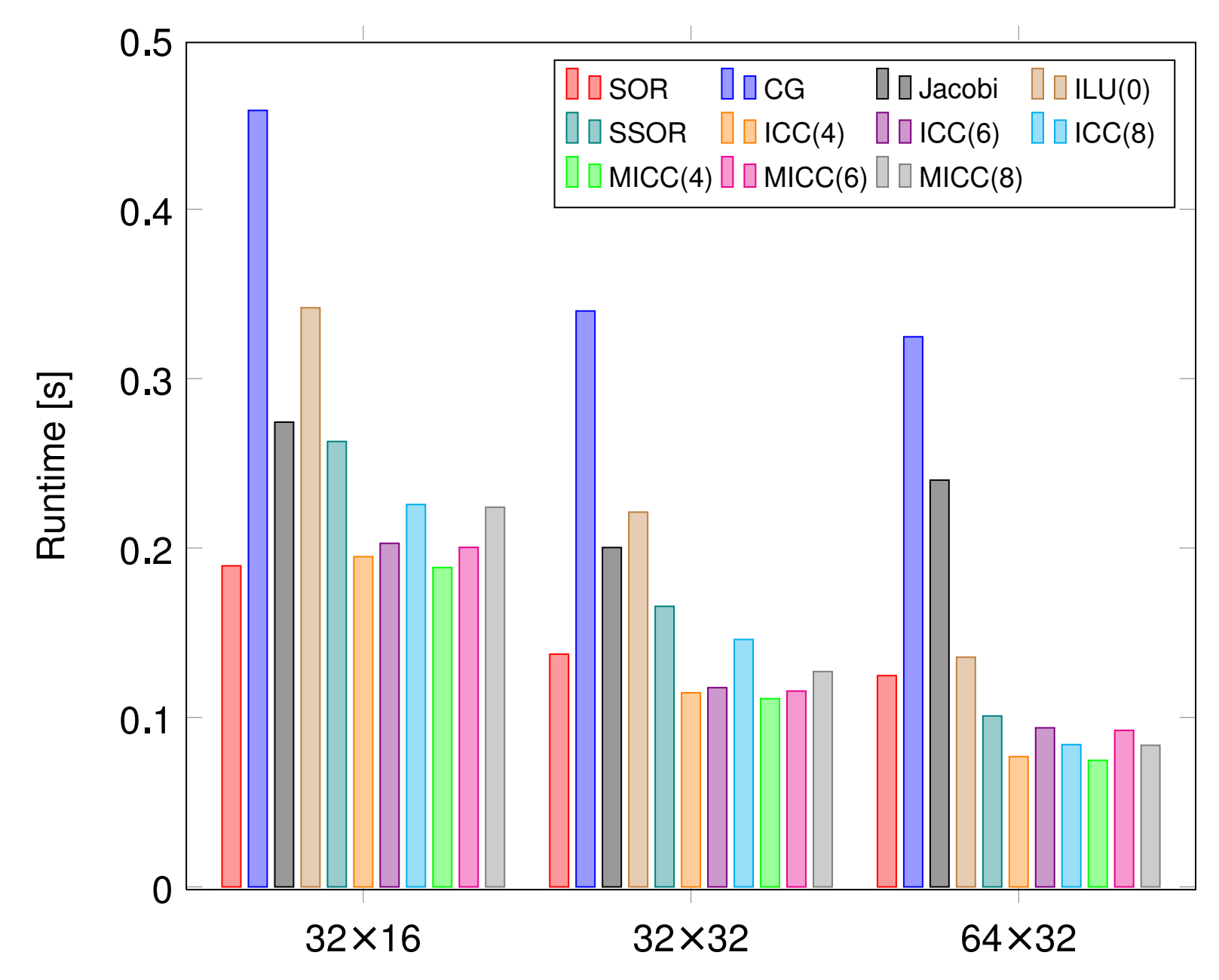

Figure: Runtime for the CG solver with multiple preconditioners in comparison to the reference SOR solver in MPIOM

CDI-pio

Design goals for CDI with parallel I/O

- Asynchronous I/O with respect to computation
- Backwards compatibility
- Small changes to CDI's API
- Set-up options for different architectures, filesystems and MPI implementations
- Best performance and scalability on DKRZ Blizzard (Power6, AIX 6.1)
- Write on multiple hardware nodes

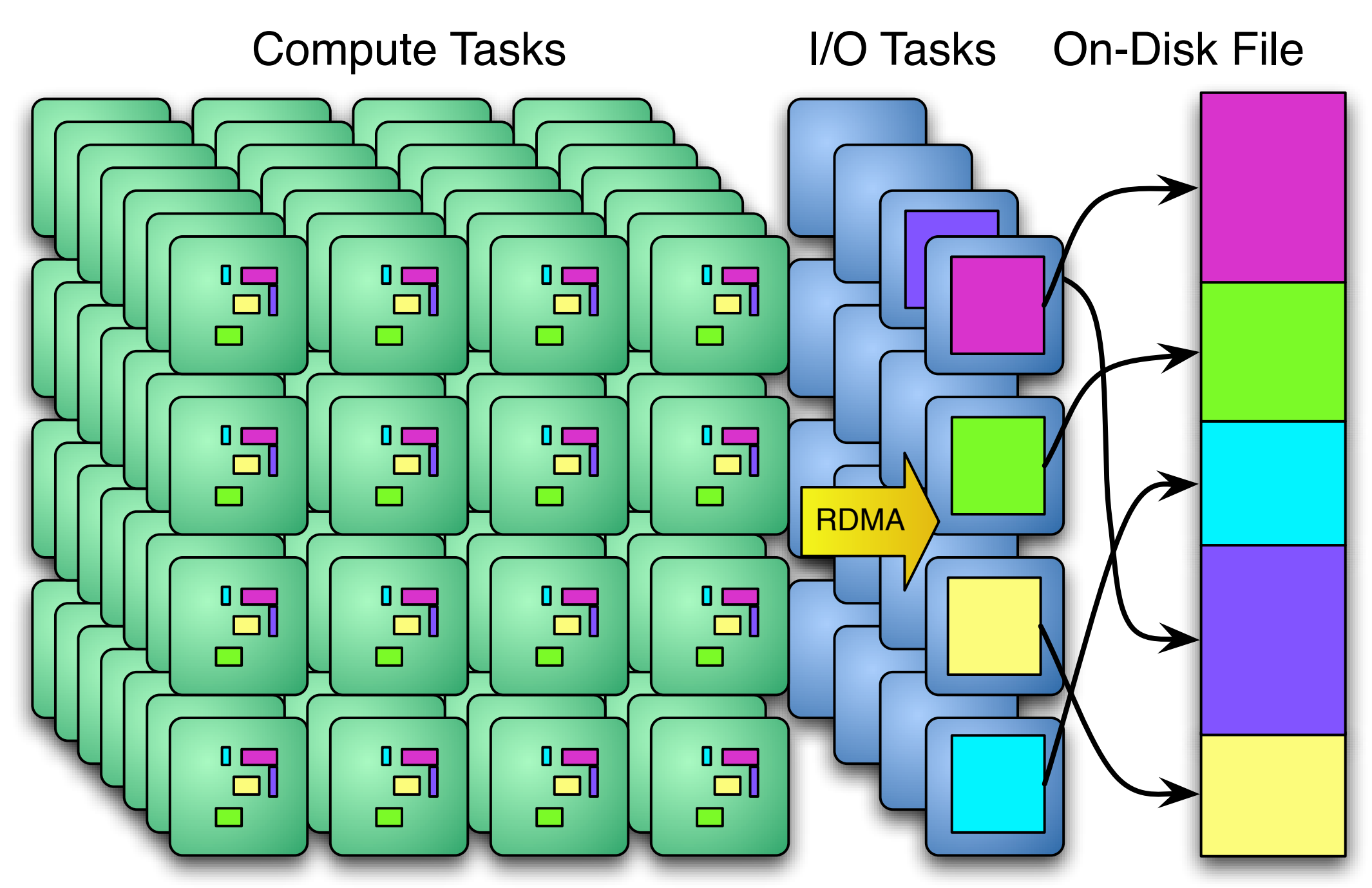

Figure: CDI supplements the processes carrying out the model computation with separate offload tasks for I/O

Improved Gather

A naive approach implements only the unavoidable data transfer from all processes to the target. This does not account for the overhead of inserting subarray data into the array. This overhead is concentrated at the target.

Our approach tries to reduce the overhead by assembling intermediate subarrays which possess the contiguous property within the target's array. These can be inserted directly using a collective mpi routine. The assembly of these subarrays can be done in parallel. Thus the overhead is distributed to several processes whose optimal number depends on the latency of the communication.

This results in a two-phase gather that, e.g. for a 192x96x47 double precision array distributed over 8x4 processes, is one order of magnitude faster than the naive approach (measured on IBM Power6 p565 running AIX). With higher parallelization one can further improve performance by adapting the procedure to the usually nonuniform communication properties of the interconnect, e.g., shared memory communication for the first phase and InfiniBand communication for the second.

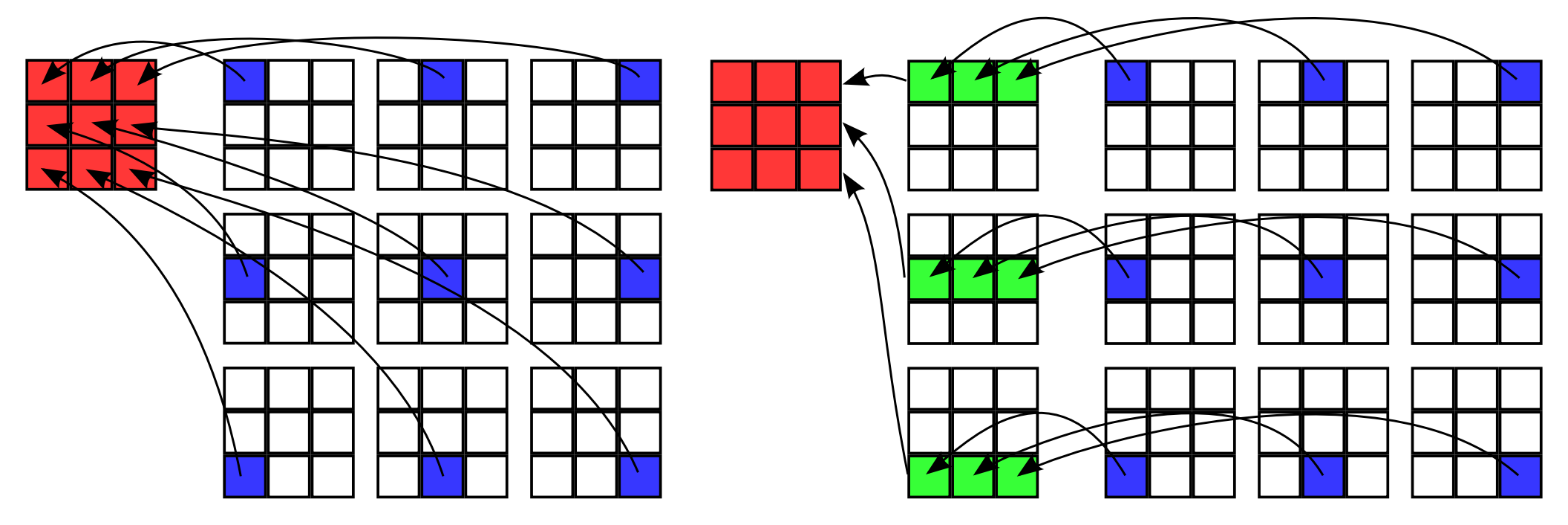

Figure: Simple gathering compared to a 2-phase implementation

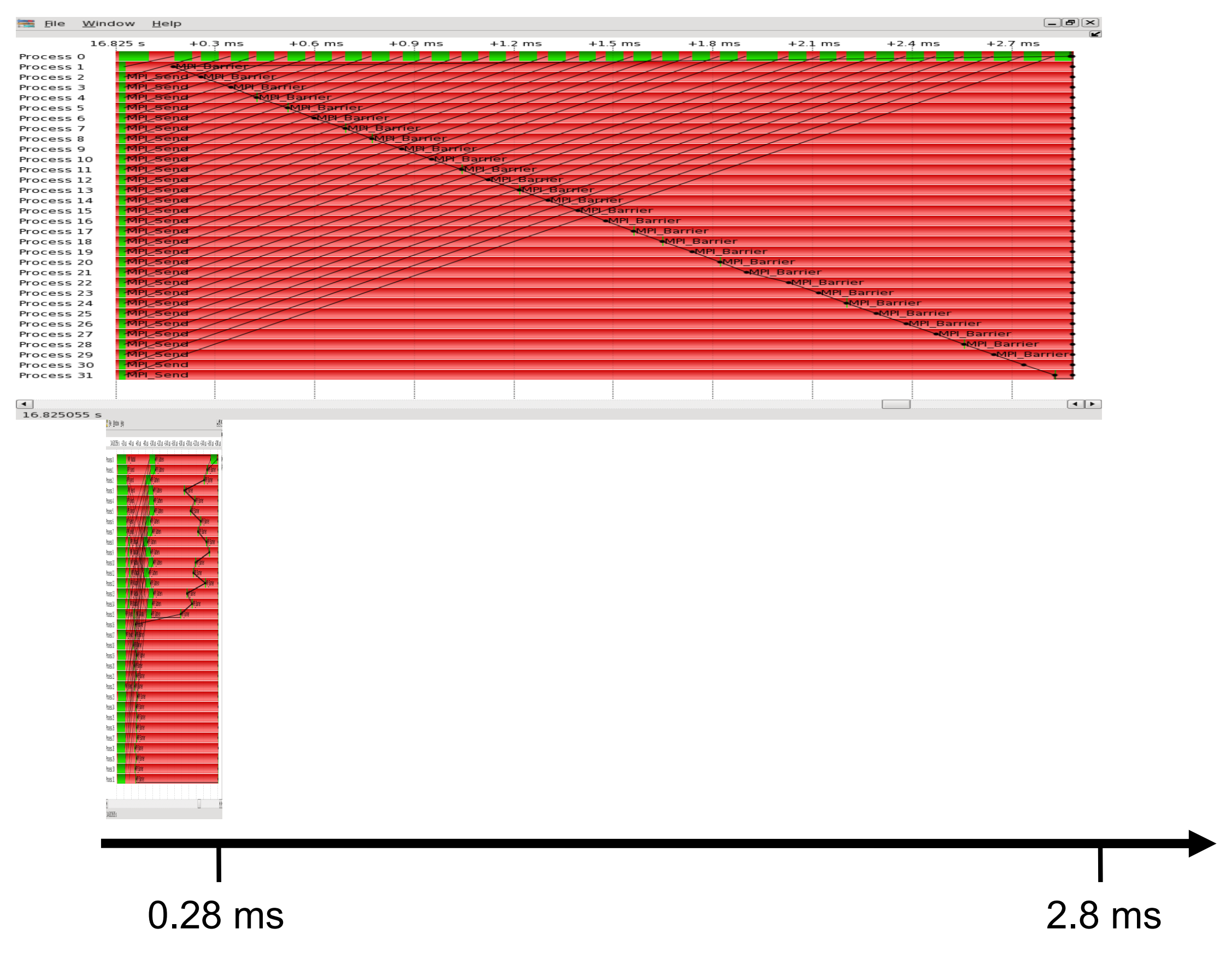

Figure: Run-time for gather decreases by a factor of 10 when comparing naïve and 2-phase gather

YAXT

The traditional approach to domain decomposition in model software via direct MPI calls incurs too much complexity, too many errors and results in low flexibility because of the low abstraction level MPI provides. **Unitrans** was conceived as a tool to describe arbitrarily decomposed enumerable data structures. This initial Fortran-only implementation showed problems because of inherent limitions of the Fortran type model and some simplifications, which only became apparent in the course of the initial implementation. Thus the concept was reimplemented in the C programming language which allows for type-agnostic functions under the name **YAXT**.

Main concepts of **Unitrans** and **YAXT**:

- Decompositions
  - Describe the logical decomposition of global data
    - By means of a distributed index
  - Type agnostic
    - reals, integers, arrays, cows and flying saucers…
- Local Data representations
  - adds physical type and storage information
    - via offset arrays and complex type descriptors
- Transposition Templates
  - Describes the logical exchange pattern between two different decompositions
    - Via distributed communication maps
- Transposition Instances
  - executable plans describing how to perform a transposition between two local data representations in different decompositions

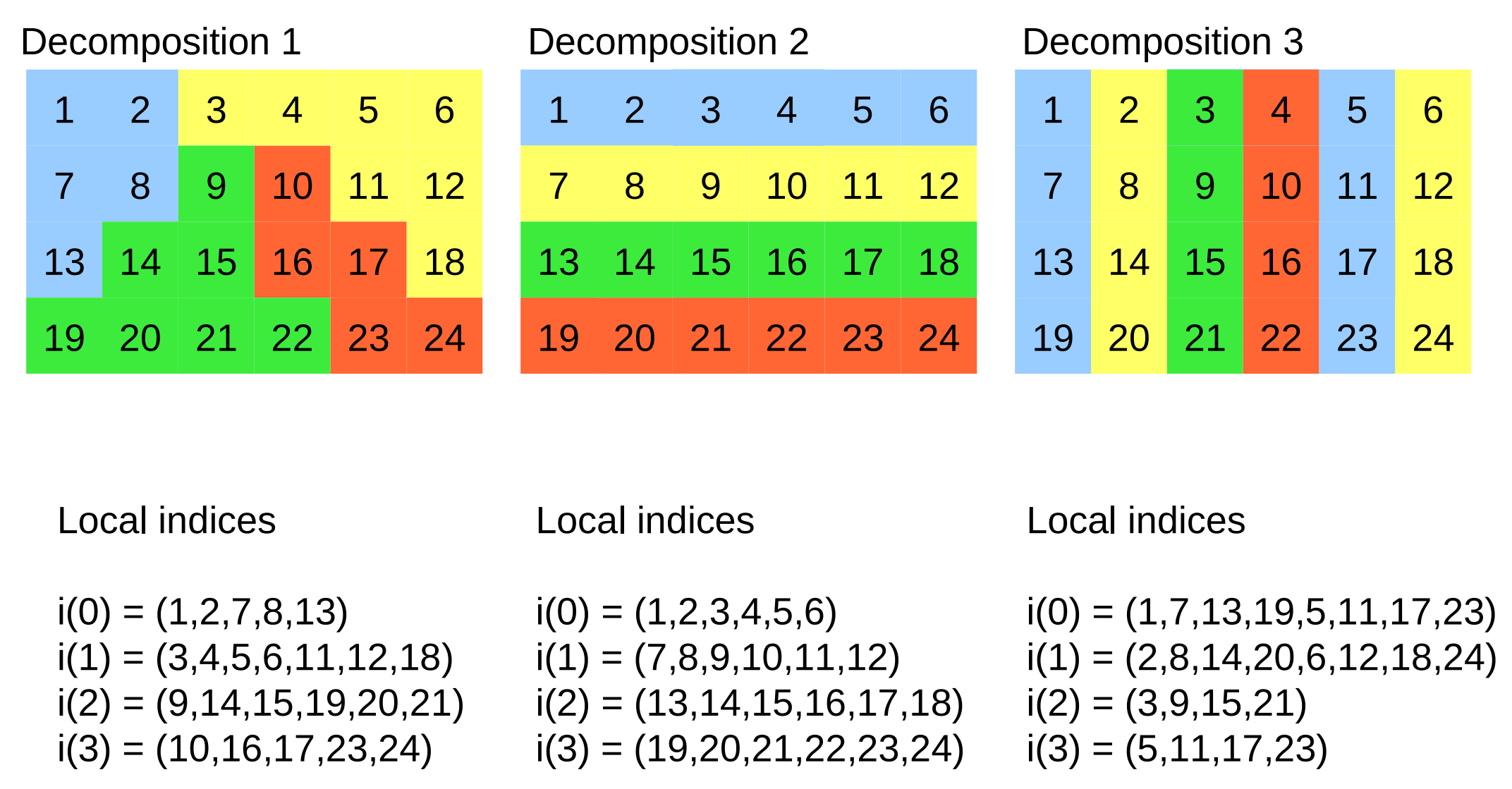

Figure: Examples of local index descriptions, can be computed with partitioning functions from PPM

(Re-)partitioning with ScalES-PPM

Using the default ECHAM decomposition, load-imbalance in MESSy/EMAC at medium scale is severe and the run-time varies by a factor of 4, i.e. some MPI tasks are idle for 75% of the time. Therefore a static round-robin decomposition for grid-point data (which includes all chemical species) was implemented that balances chemistry computations well for $\frac{N}{P} \geq 32$ where N is the number of grid-points and P the number of processes, but this only results in run-time benefits when additionally using the advanced communication methods of **Unitrans**.

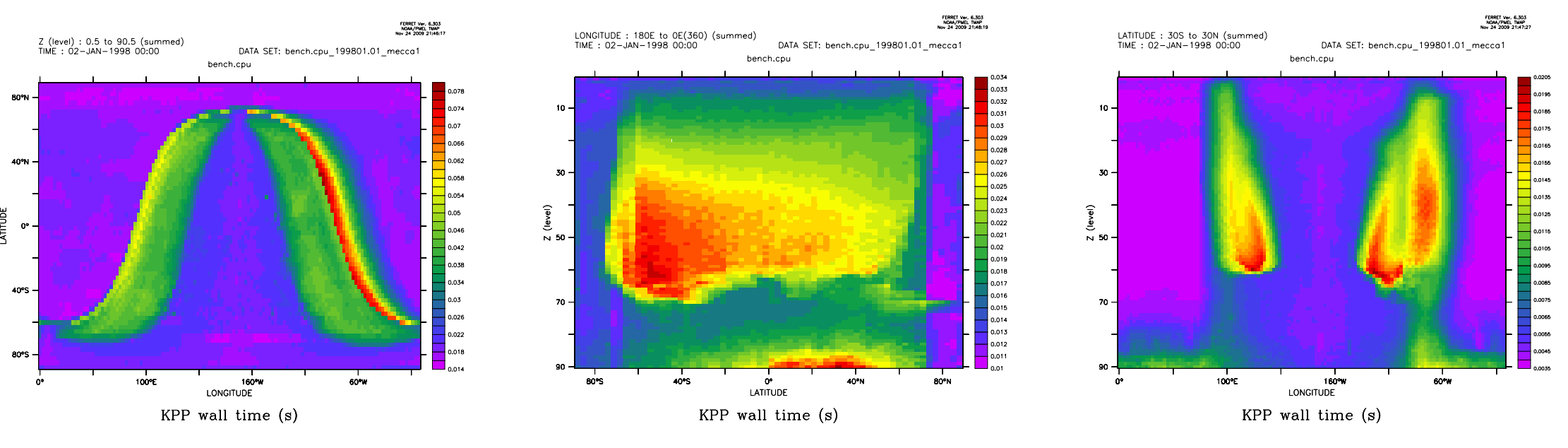

Figure: The vertical, longitudinal and latitudinal sums of the KPP solver run time show that the imbalance is correlated with zones of sunrise and sunset.

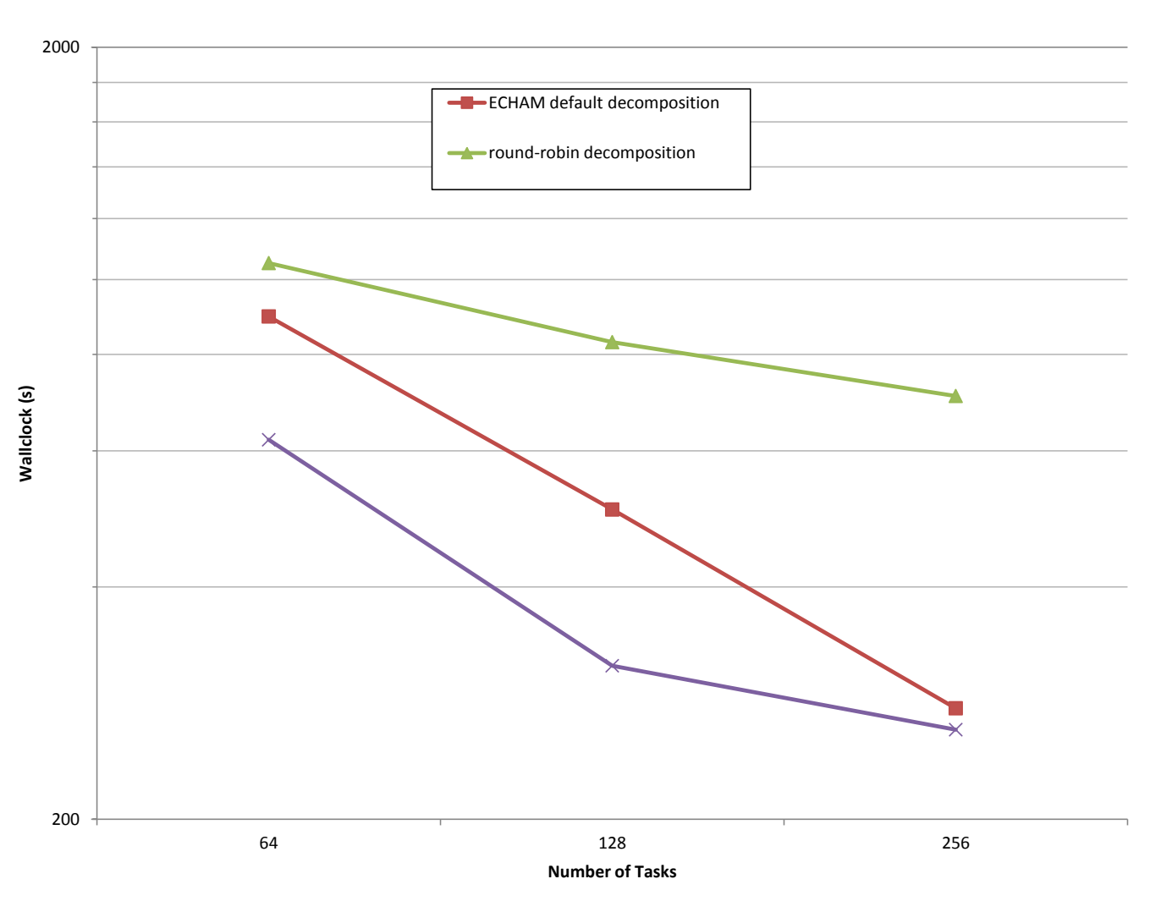

Figure: Load-balanced round-robin decomposition and Unitrans communication result in improved run-time for EMAC T42L90MA on DKRZ blizzard with SMT mode

To go to even higher scale, a dynamic parallel repartitioning scheme was implemented that swaps computationally expensive for cheap domain parts that keeps the number of elements per task constant (thus removing the need for reallocations). It has demonstrated performance increases for $\frac{N}{P} < 32$.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
DLR Projektträger
Förderkennzeichen: 01IH08004