CONSORTIUM FOR SMALL SCALE MODELING

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

COSMO, ICON and Computers

Ulrich Schättler

Deutscher Wetterdienst

Research and Development

Contents

Problems of the COSMO-Model on HPC Architectures

(Recall from Exeter, 2010)

Some HPC Facts

- **Massive concurrency** – increase in number of cores, stagnant or decreasing clock frequency
- **Less and "slower" memory per thread** – memory bandwidth per instruction/second and thread will decrease, more complex memory hierarchies
- **Only slow improvements of inter-processor and inter-thread communication** – interconnect bandwidth will improve only slowly
- **Stagnant I/O sub-systems** – technology for long-term data storage will stagnate compared to compute performance
- **Resilience and fault tolerance** – mean time to failure of massively parallel system may be short as compared to time to solution of simulation, need fault tolerant software layers

We will have to adapt our codes to exploit the power of future HPC architectures!

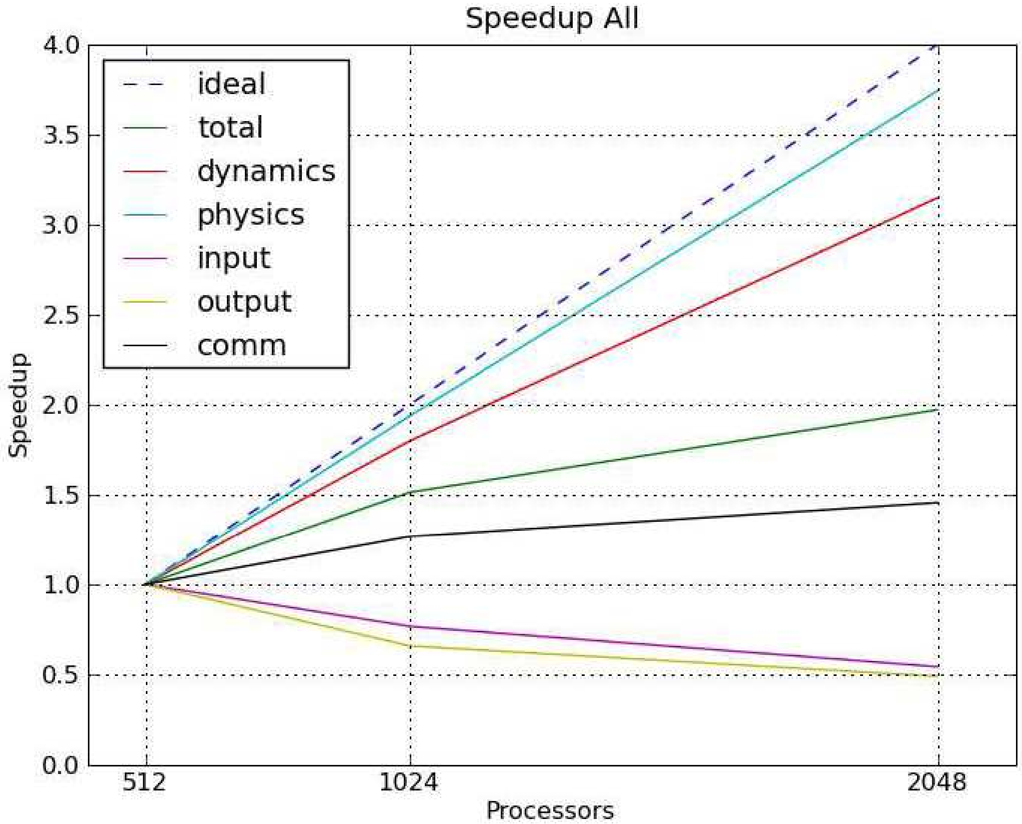
Problems on existing Computers

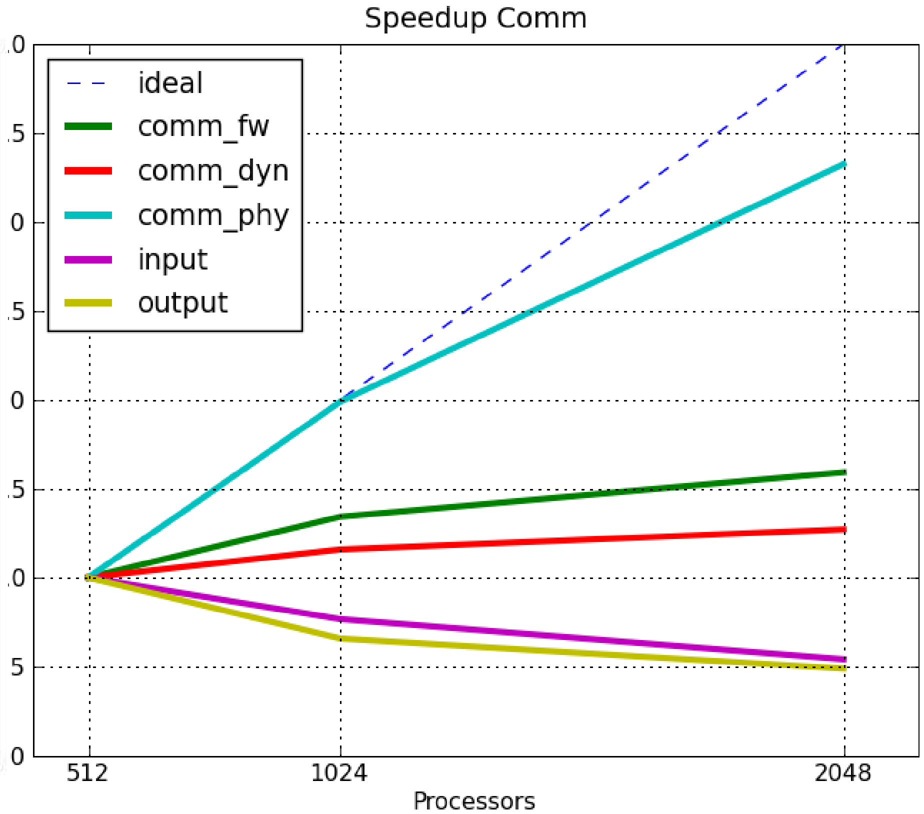
- 21 hours COSMO-DE forecast

	NEC SX-9 8 Procs	IBM pwr6 256 procs
Computations Dynamics	729.59	570.44
Computations Physics	506.18	220.45
Communications	115.61	207.69
I/O	124.43	108.40
% of I/O and Comm.	15	25

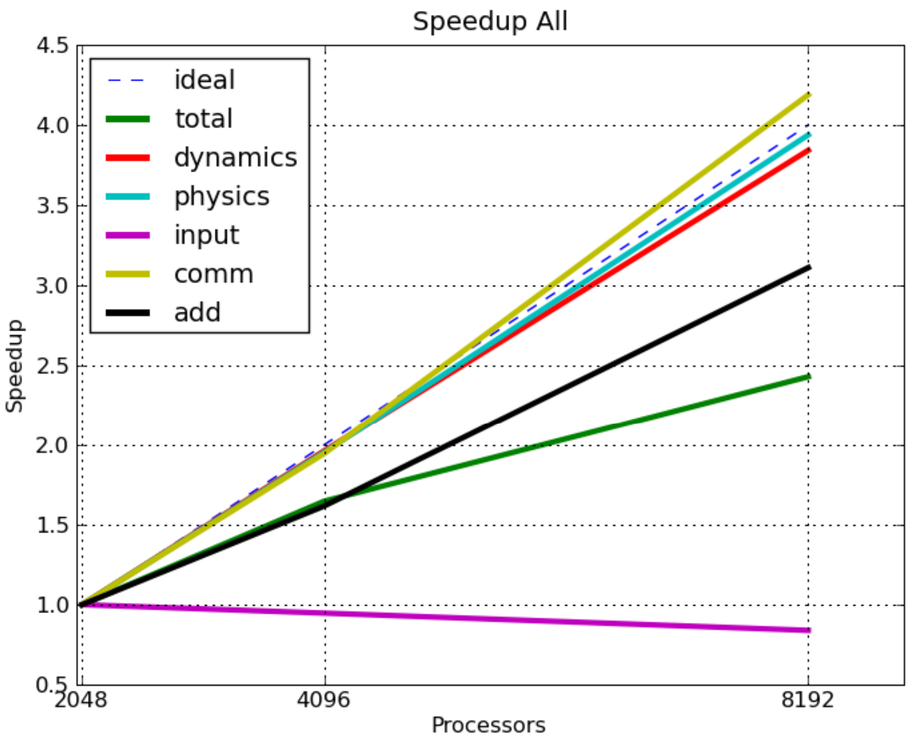
- Code efficiency
  - NEC SX-9: 13 % of peak
  - IBM pwr6: about 5-6 % of peak
  - Cray XT4: about 2-3 % of peak

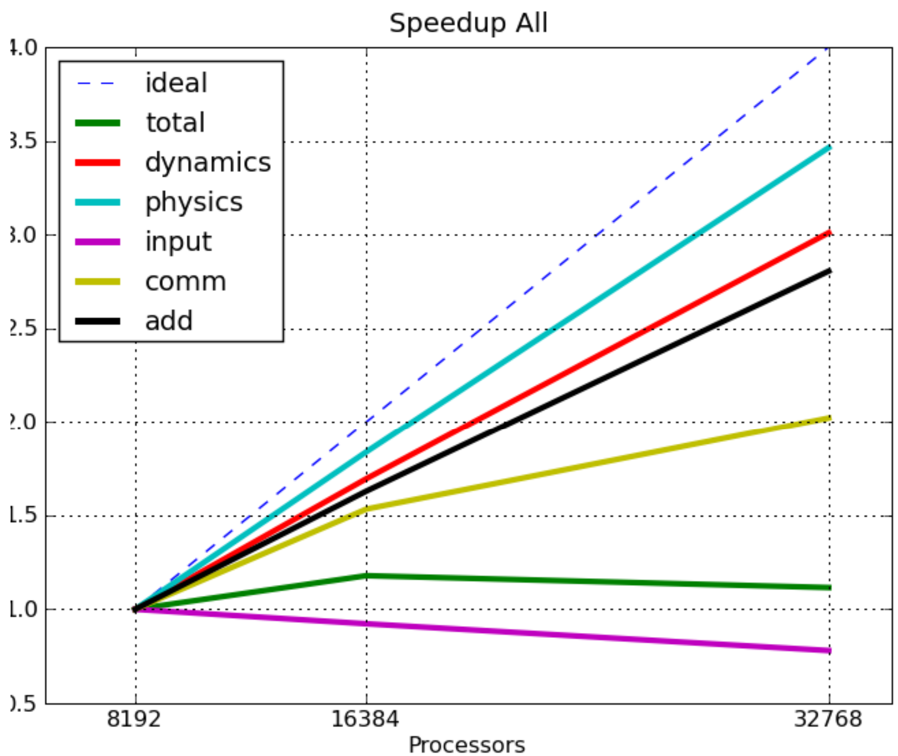
Scalability of COSMO-DE: 421 × 461 × 50, 21h





Scalability of COSMO-Europe: 1500 × 1500 × 50, 2.8 km, 3 h, no output





The Problems of the COSMO-Code …

- I/O: Accessing the disks and the global communication involved disturbes scalability heavily.
- Although the communications besides I/O are almost all local, the speedup degrades when using many processors.
- What cannot be seen on the pictures before: Although the speedup of the computations is not bad, the efficiency of the code is not satisfying:
  - NEC SX-9: 13 % of peak
  - IBM pwr6: about 5-6 % of peak
  - Cray XT4: about 2-3 % of peak
- This is because of the memory boundedness of the code
- Fault Tolerance: Besides „Restart-Files“ (model checkpointing and restart) there are no means to care for hardware failures. But writing restarts also is very expensive.

HP2C and POMPA

HP2C project COSMO-CLM

- Swiss Initiative: High Performance High Productivity Computing
- "Regional Climate and Weather Modeling on the Next Generations High-Performance Computers: Towards Cloud-Resolving Simulations"
- **Tasks**
  1) Cloud resolving climate simulations (IPCC AR5)
  2) Adapt and improve existing code (improved communications, hybrid parallelization, I/O)
  3) Rewrite of dynamical core
- **Funding** ~ 900 kCHF, 3 years, 6 FTEs + core group

COSMO PP POMPA (Lead: Oli Fuhrer)

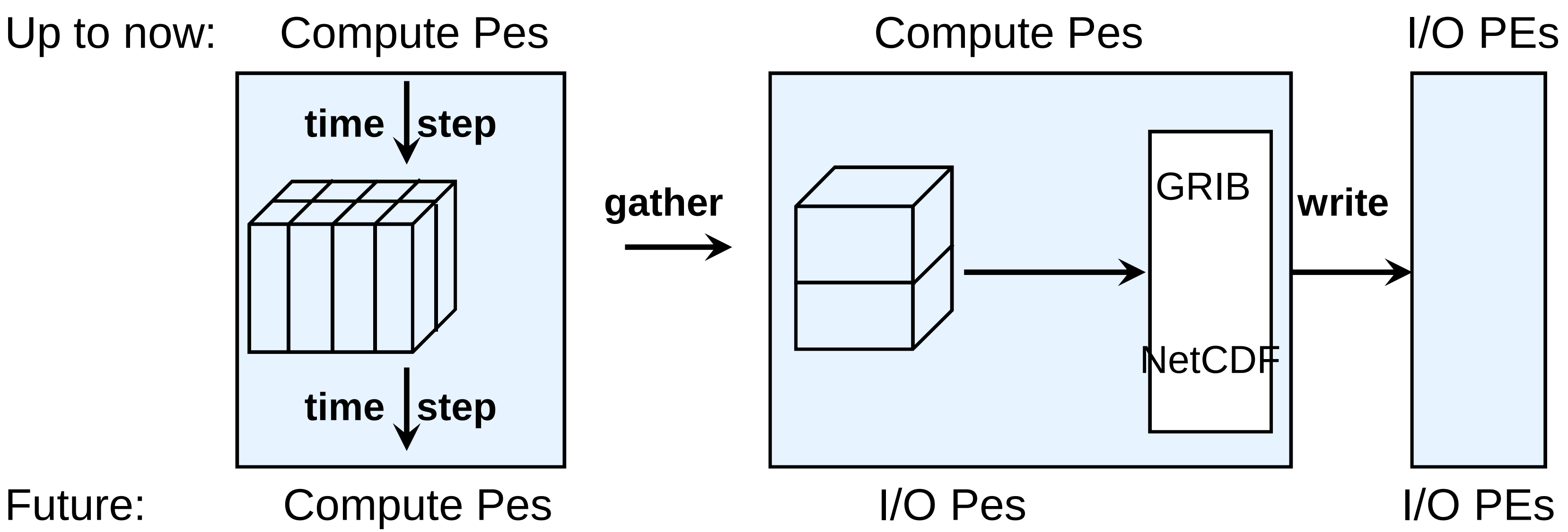
- **P**erformance **O**n **M**assively **P**arallel **A**rchitectures
- Timeframe of 3 years (Sept. 2010 – Sept. 2013)
- **Goal:** Prepare COSMO code for emerging massively parallel architectures, especially help in implementing HP2C work into the official COSMO code
- Tasks overview:
  - Do a performance analysis
  - Check and improve MPI communications
  - Try hybrid Parallelization: Can that improve scaling?
  - Tackle the I/O bottleneck: check existing asynchronous I/O; investigate parallel I/O
  - Explore GPU acceleration and possibilities of simple porting of parts of the code to GPUs
  - Redesign of the dynamical core: design a modern implementation of the dynamical core that maps more optimally onto emerging architectures

COSMO PP POMPA: Status of Work (I)

- ➔ Performance Analysis (A. Roches, J-G. Piccinalli, O. Fuhrer et al.)
  - ➔ has been done, but did not detect other than the “usual suspects”
- ➔ MPI communications (Stefano Zampini, CASPUR / CNMCA)
  - ➔ tried non-blocking halo exchange and collective communication
  - ➔ can be done, but bigger changes in the code are necessary to really overlap communication and computation.
- ➔ Hybrid Parallelization (Stefano Zampini, CASPUR; Matt Cordery, CSCS)
  - ➔ has been done for Leapfrog and Runge-Kutta dynamics on the loop level, but did only reach same performance as pure MPI implementation
  - ➔ again, more changes to the code are necessary, to gain performance with a hybrid MPI / OpenMP implementation

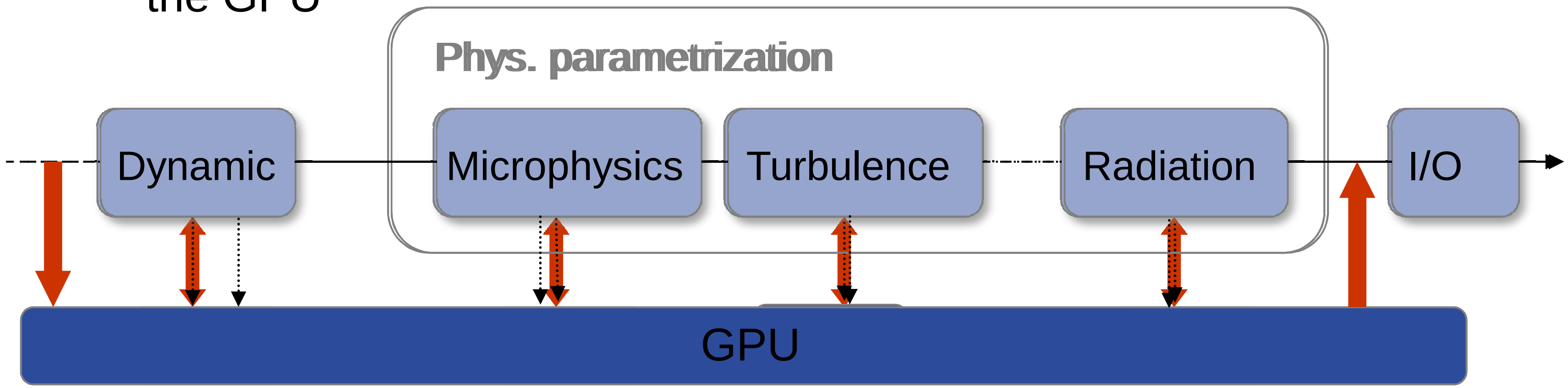
COSMO PP POMPA: Status of Work (II)

- Tackle the I/O bottleneck: check existing asynchronous I/O; investigate parallel I/O (Neil Stringfellow et al., CSCS)
  - Tests with parallel NetCDF on O(1000) cores showed problems with non-scalable meta data (opening a file on a modern parallel file system is not a scalable operation)
  - have to investigate more sophisticated asynchronous strategies



COSMO PP POMPA: Status of Work (III)

- Explore GPU acceleration and possibilities of simple porting of parts of the code to GPUs (Xavier Lapillonne, MeteoSwiss)
  - Speed up of microphysics using Fermi card and double precision reals with respect to reference MPI CPU code running on 6 cores Opteron:
    - 10x without data transfer
    - 2x when considering data transfer
  - Because of large overhead of data transfer going to GPU is only viable if more computation is done (i.e all physics or all physics + dynamics) on the GPU



COSMO PP POMPA: Status of Work (IV)

- Redesign of the dynamical core: design a modern implementation of the dynamical core that maps more optimally onto emerging architectures (SCS, Zürich)
  - memory bandwidth is the main performance limiter on commodity hardware
  - have to check memory layout and implementation of operators to get an improvement
  - Plan: develop a DSEL (domain specific embedded language) like „stencil-library“, where implementation of operators is highly optimized.
- Fully functional single-node CPU implementation in C++ available
  - fast wave solver, horizontal advection (5^{th}-order upstream, Bott), implicit vertical diffusion and advection, horizontal hyper-diffusion, Coriolis and other stencils
- Verified against Fortran reference to machine precision

COSMO PP POMPA: Status of Work (V)

- Performance of the prototype is promising

Domain Size	COSMO	Rewrite	Speedup
32x48	19.06 s	10.25 s	**1.86**
48x32	16.70 s	10.17 s	**1.64**
96x16	15.60 s	10.13 s	**1.54**

- But: Usage of C++ not yet decided
- Is a stencil-library and way to implement the dynamics acceptable by the developers?
- How much of the performance gain is due to C++ and how much is due to better memory layout and optimized operator implementation?
- Even if we go that way, an operational implementation of the full dynamical core will take more time

COSMO PP POMPA: Early Experiences

- Quick improvements with modest code changes could not be done
  - There is not always a free lunch!
  - We still think that improvements are possible, but not developed and implemented within few weeks or months.
- More far-reaching developments
  - GPUs give a higher peak performance at lower cost / power consumption and seem to be a valid alternative to todays architectures.
  - But there are NO programming standards across different platforms (CUDA, OpenCL, directive based approaches). How long will it take to define such standards?
  - Could traditional CPUs benefit from GPU developments?
- Do we have to say „Good Bye Fortran“?

ICON

The next-generation global model at DWD and MPI-M

The following slides are courtesy of the colleagues at DWD and MPI-M

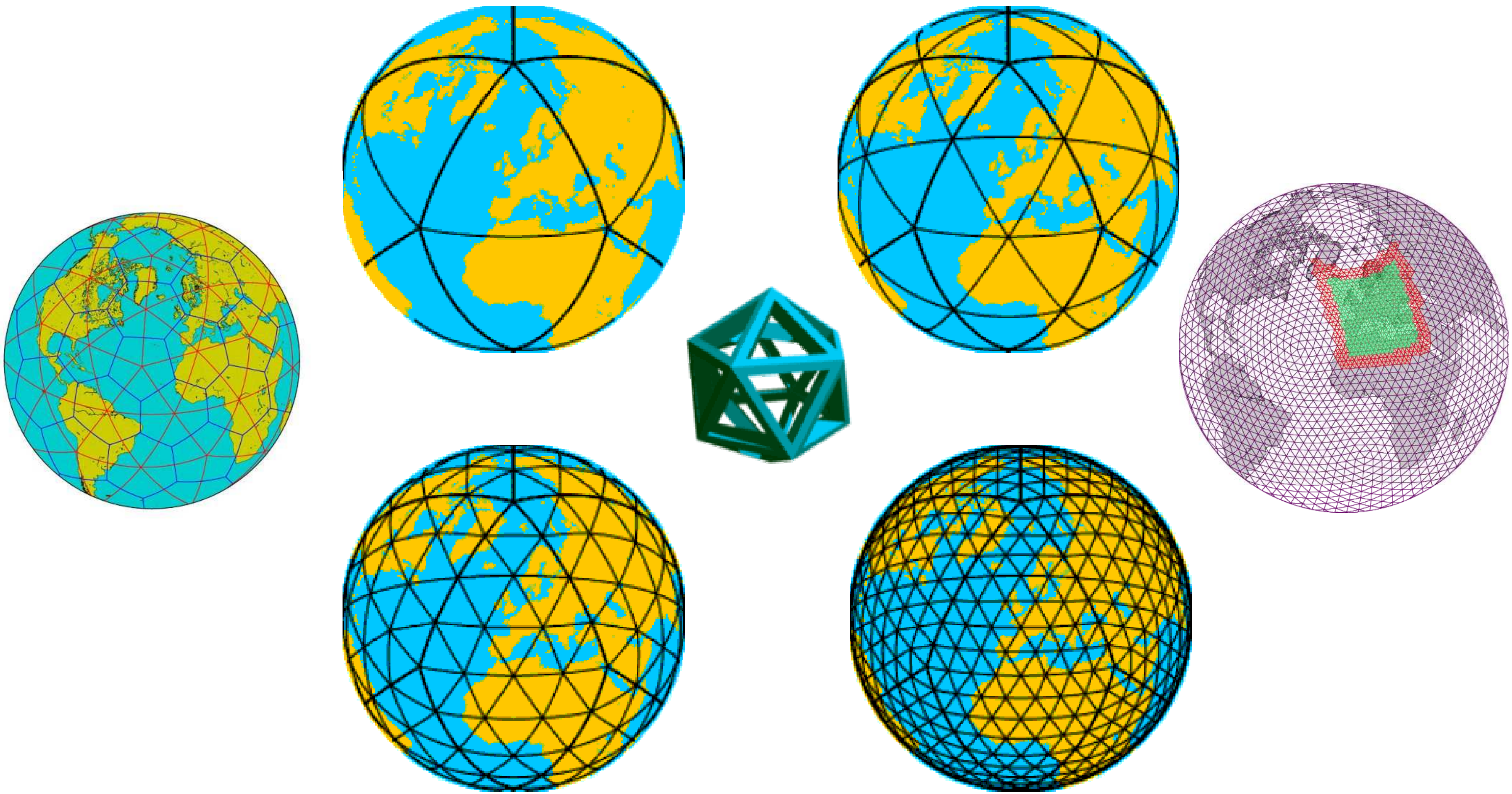
Project Teams at DWD and MPI-M

D. Majewski	**Project leader DWD** **(till 05/2010)**
G. Zängl	**Project leader DWD** **(since 06/2010)** two-way nesting, parallelization, optimization, numerics
H. Asensio	external parameters
M. Baldauf	NH-equation set
K. Fröhlich	physics parameterizations
M. Köhler	physics parameterizations
D. Liermann	post processing, preprocessing IFS2ICON
D. Reinert	advection schemes
P. Ripodas	test cases, power spectra
B. Ritter	physics parameterizations
A. Seifert	cloud microphysics
U. Schättler	software design
MetBw	
T. Reinhardt	physics parameterizations

M. Giorgetta	**Project leader MPI-M**
M. Esch	software maintenance
A. Gaßmann	NH-equations, numerics
P. Korn	ocean model
L. Kornblueh	software design, hpc
L. Linardakis	parallelization, grid generators
S. Lorenz	ocean model
C. Mosley	regionalization
R. Müller	pre- and postprocessing
T. Raddatz	external parameters
F. Rauser	adjoint version of the SWM
W. Sauf	Automated testing (Buildbot)
U. Schulzweida	external post processing (CDO)
H. Wan	3D hydrostatic model version

External: *R. Johanni*: MPI-Parallelization

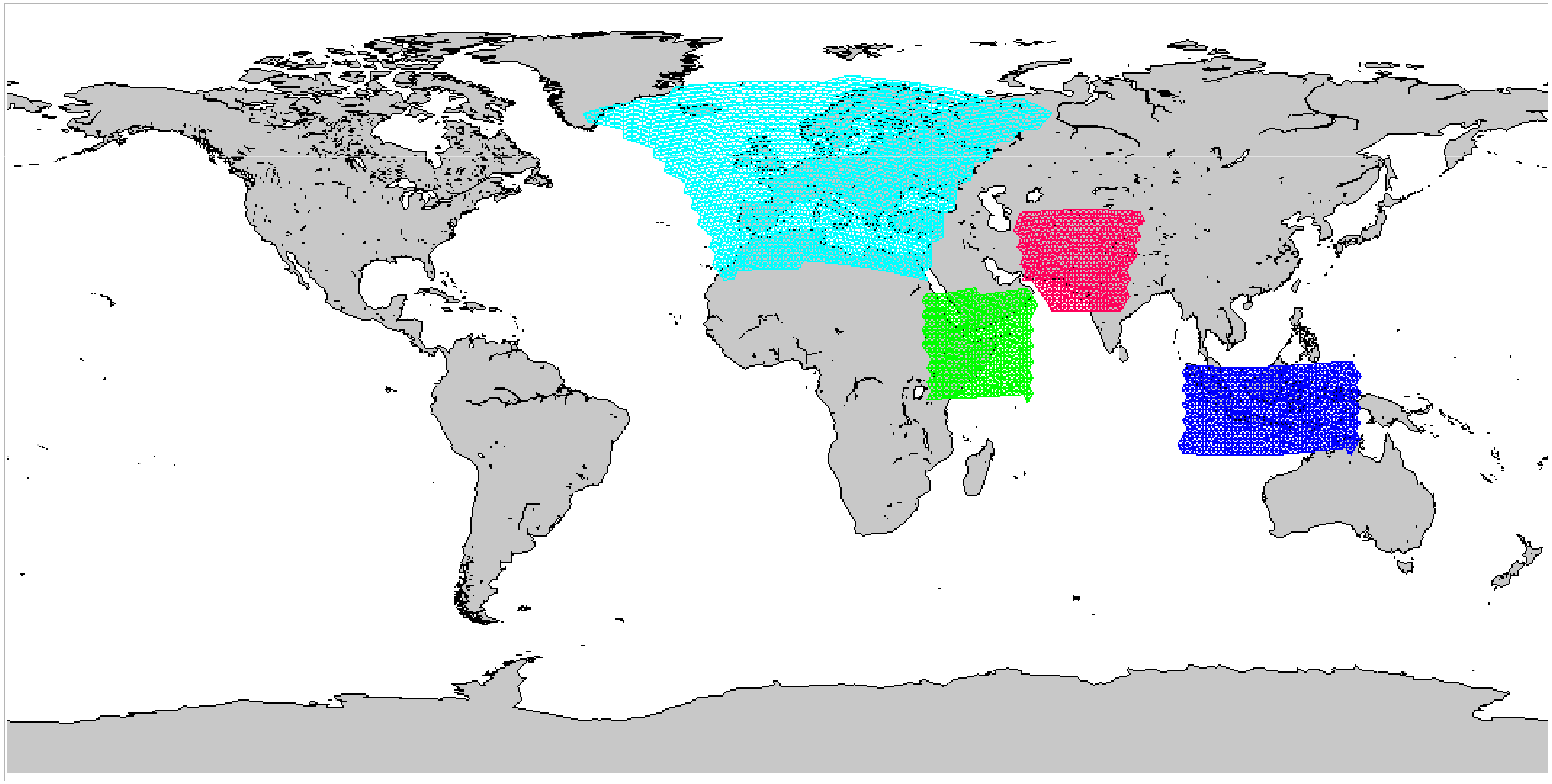
The Horizontal Grid



CONSORTIUM FOR SMALL SCALE MODELING COSMO

Deutscher Wetterdienst
Wetter und Klima aus einer Hand

Example for domain configuration with multiple nests



Combining several domains at the same nesting level into one logical domain significantly reduces the parallelization overhead

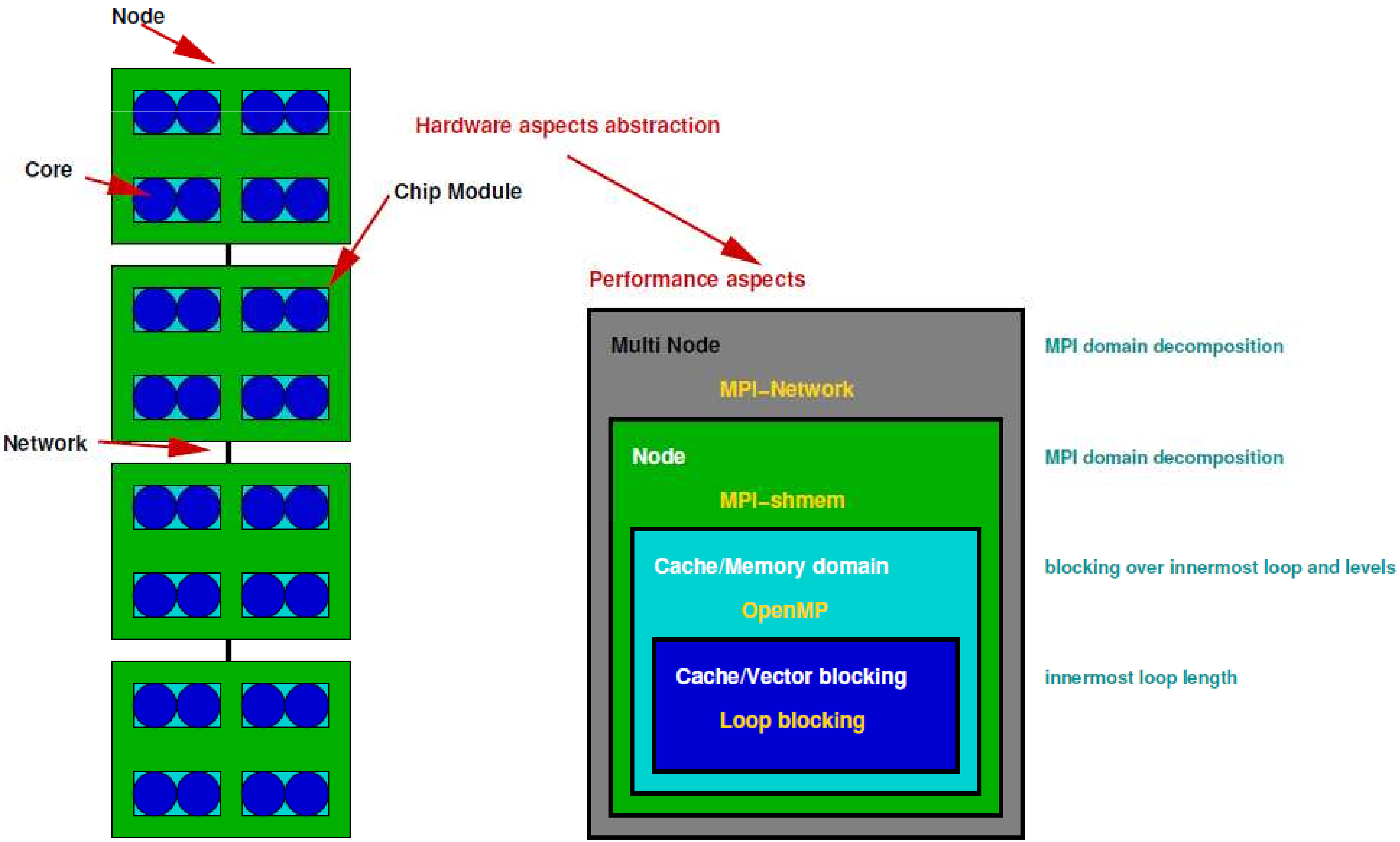
General Coding Structure

- **Memory layout looks at horizontal grid as being unstructured `(ngridpoints, nlevel)` ⇒ differencing and averaging operators involve indirect addressing**
- **Selectable inner loop length (“`nproma`”-blocking)**
- **Long outer DO loops to optimize cache use and to minimize the number of OpenMP sections**
- **Tracer variables (moisture, clouds, precipitation etc.) are stored in a 4D array; operations common to all tracers are done in one step**
- **Physics parameterizations are called in 2D slices (`nproma*nlevel`), OpenMP parallelization is done outside the physics schemes**

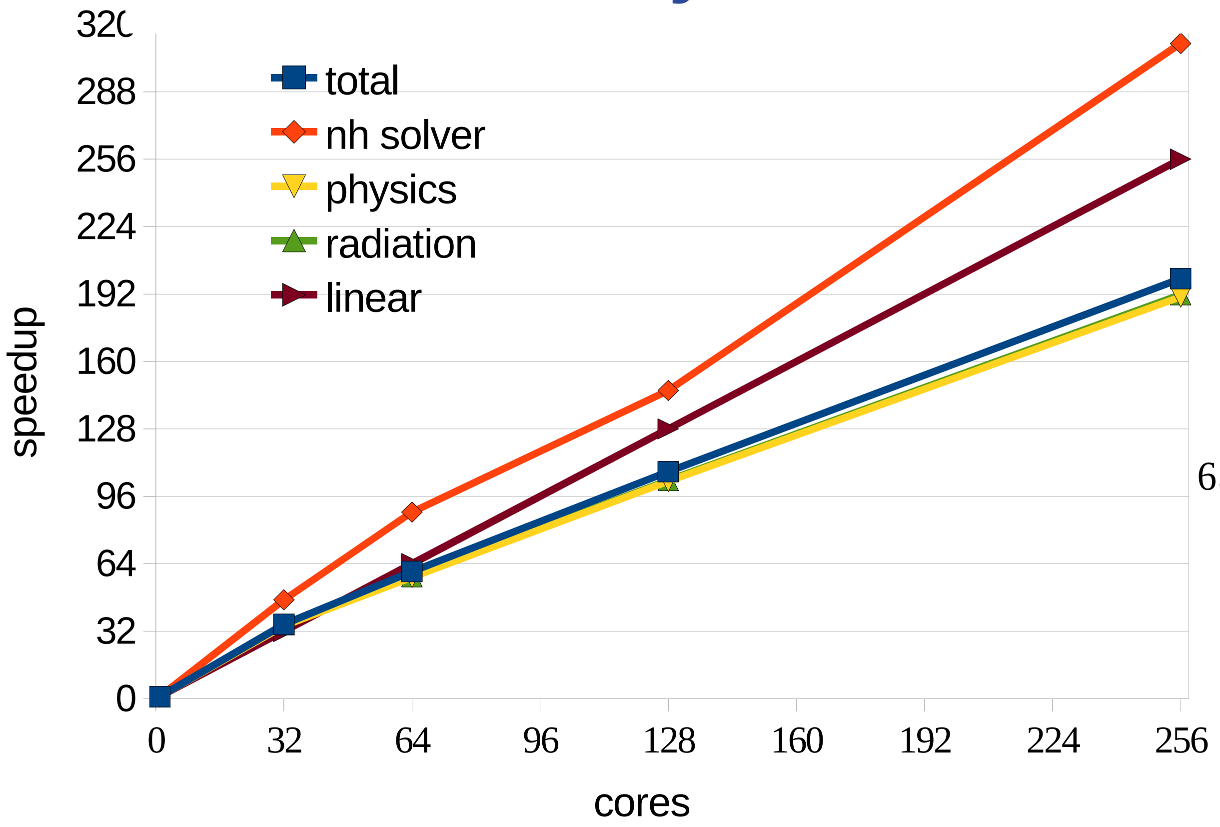
Aspects for Parallelization

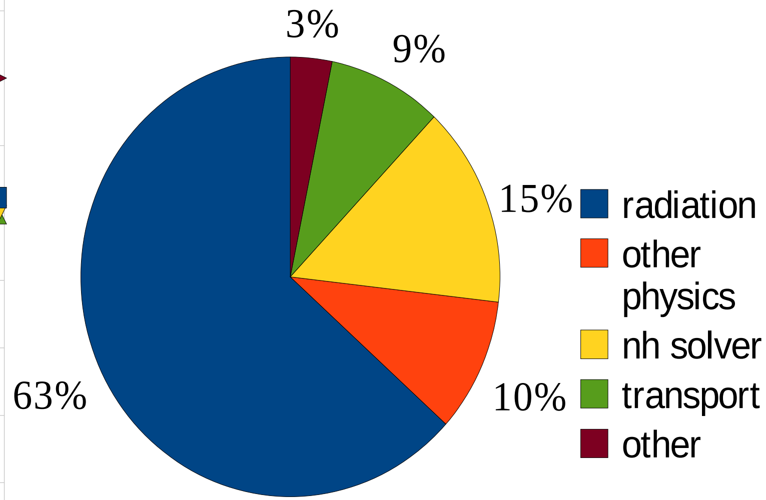
- Minimize number of communication calls; if more than one variable requires halo exchange at the same time, the related send/receive calls are combined into one
- Within a subdomain, first the points to be exchanged for the halo update are computed, then the interior points. This (potentially) allows overlap of communication and computations.
- If one-way and two-way nesting are combined (which prevents combining all nests into one logical domain), both groups of nests can be processed in parallel on a suitably chosen subset of processors (processor splitting)
- Parallel asynchronous I/O with a selectable number of I/O processors

Levels of Parallelism



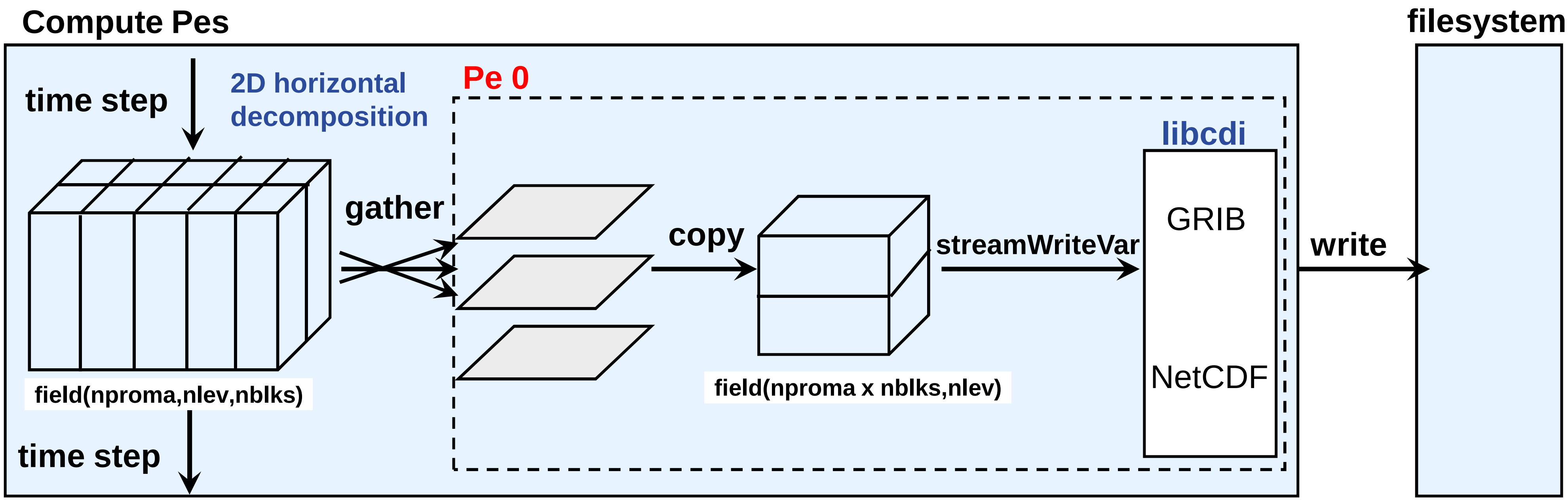
First Scalability Results





- ➔ **Single domain, 81920 cells (~70 km res.), 35 levels, IBM Power6**
- ➔ **Super-linear scaling for dynamical core (better cache use)**
- ➔ **Sub-linear scaling for physics schemes (halo points)**

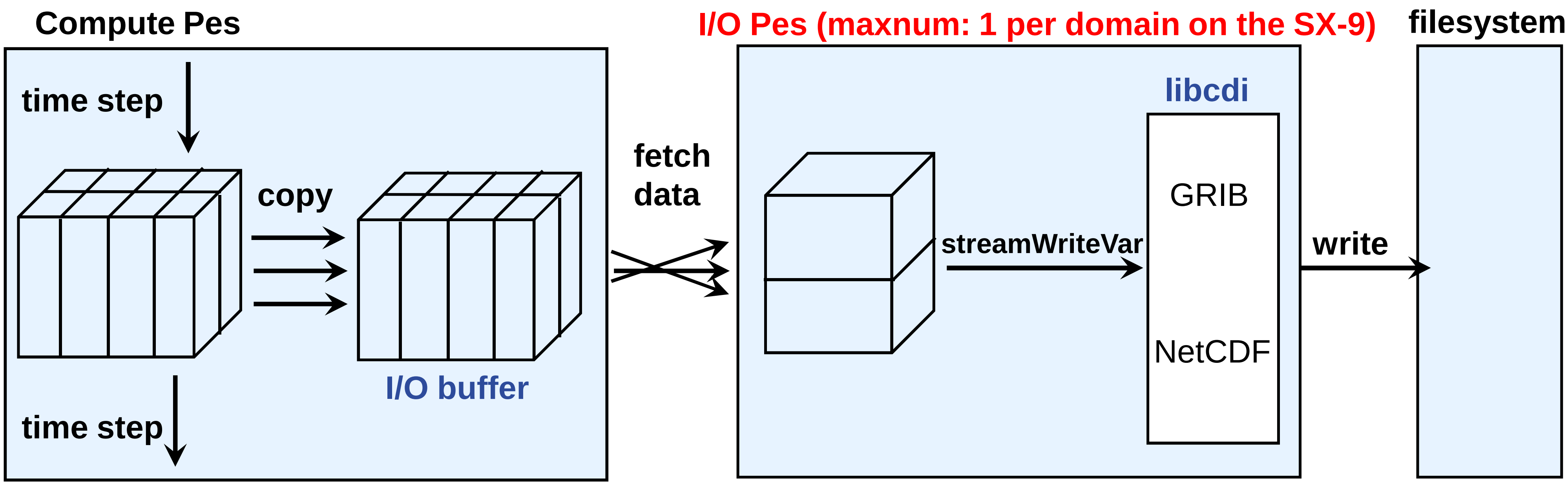
Synchronous I/O



Problem

All processes wait until process 0 has written the data

Asynchronous I/O

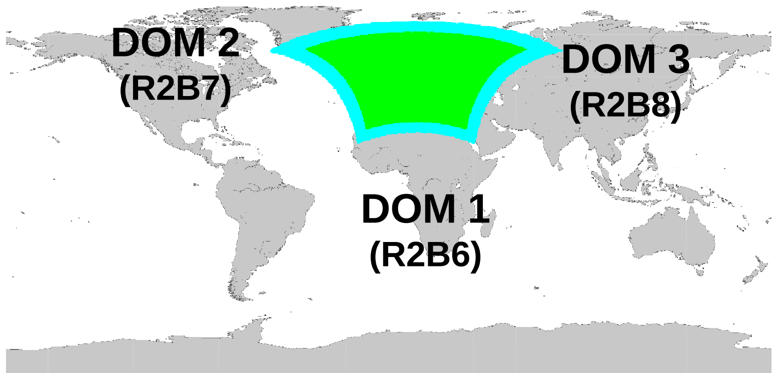


- After each I/O timestep the compute processes copy their data to a buffer and go on calculating until the next I/O timestep.
- I/O Pes fetch the data using one sided MPI communication.
- Every I/O Pe gets at least 1 domain (1 output file per domain)

Permits computations to continue while I/O is done by dedicated I/O Pes

Asynchronous I/O on NEC SX-9

- **Setup:**
- Global domain with two nested domains
- Simulation time 7 days
- 1 output file for each domain and output time
- Total data amount of 144/37/96 = 277 GB
- Runs with 12 compute Pes and 0,1,2, or 3 I/O Pes



Runtime

	0 I/O Pes	**1 I/O Pe**	**2 I/O Pes**	**3 I/O Pes**
Real time [s] (output every 12h)	3525.2	3415.5 (-3.11%)	3409.5 (-3.28%)	3473.6 (-1.46%)
Real time [s] (output every hour)	4440.2	3464.4 (-22.0%)	3442.4 (-22.5%)	3527.5 (-20.5%)

CONSORTIUM FOR SMALL SCALE MODELING COSMO

Deutscher Wetterdienst
Wetter und Klima aus einer Hand
DWD

Conclusions

- Vendors do have some ideas what to do about the computational bottleneck. They propose some kind of „Accelerators".
- At the moment there are no standards whatsoever: about the accelerators, how to use them and about the way how to program them.
- COSMO is able to test various things because of the HP2C project.
- Will we be able to set some standards?
- Will we need these accelerators? Or will the CPUs take over all the good things from accelerators?

Please tune in again
in the future, when we
know more about that