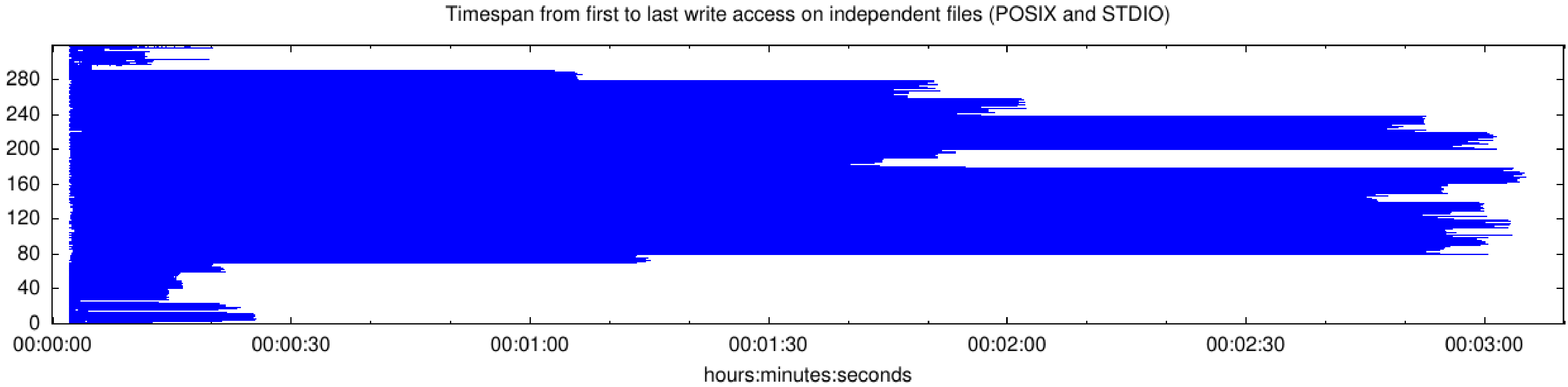
Karlsruhe Institute of Technology

Accelerating I/O in ESMs using on demand filesystems

Stefan Versick, Thomas Fischer, Ole Kirner, Tobias Meisel, Jörg Meyer



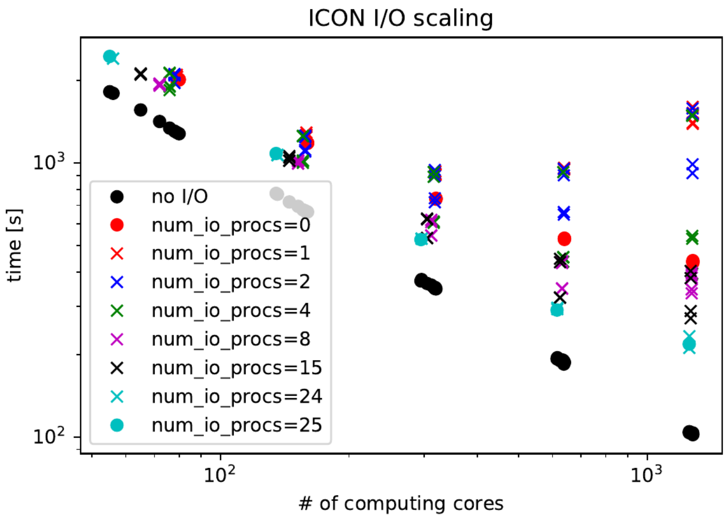
The Problem

Karlsruhe Institute of Technology

- Earth System Models (ESM) got much **more demanding** over the last years
  - Modelled processes got **more complex** and more and **more processes** are considered in models
  - **Higher resolutions** of the models are used to improve weather and climate forecasts
- This **requires** faster high performance computers (HPC), better parallization and **better I/O performance**
- Especially **scaling of I/O** with higher amounts of used cores often is a **bottleneck**

Scaling Example

- Model performance of **ICON 2.5** for different settings
- **Real world example**: original setting was num_io_procs=1 (red crosses)
- Shown is the **complete time** for the model (computations + I/O)
- In this extreme example just by changing some settings the model got almost a **factor of 10 faster** when using 1280 cores
- Take home message: **I/O can destroy** the otherwise really **good model performance**



On demand Filesystem

Karlsruhe Institute of Technology

- On demand filesystems **only exist during runtime** of job
- A temporary, user exclusive filesystem is created on the node local disks
  - **Node local disks** often are faster than the disks in the global storage (in our case SSDs instead of HDDs)
  - Exclusive: **Bandwidth is not shared** with other users (always same performance is expected)
  - Downside: Data at the end of the job is erased with the file system → **data has to be copied back by the user** (this is often (and in all our cases) much faster than directly doing output in the global file system)
- In our case **BeeGFS** was used as on demand filesystem
- Question: **Are ESMs faster when an on demand filesystem is used?**

Used ESMs

Karlsruhe Institute of Technology

- EMAC: Earth System Model; ECHAM MESSy Atmospheric Chemistry; Modular Earth Submodel System with ECHAM5 dynamical core
- ICON: (Icosahedral Nonhydrostatic) Modell: Modern weather and climate model by the DWD, DKRZ, KIT and others
- OGS: OpenGeoSys: scientific open source project for the development of numerical methods for the simulation of thermo-hydro-mechanical-chemical (THMC) processes in porous and fractured media

Other used tools

- Scalasca: Tool for performance analysis of parallelized software (https://www.scalasca.org/)
  - Used for identifying I/O as bottleneck
- Darshan: HPC I/O characterization tool (https://www.mcs.anl.gov/research/projects/darshan/)
  - Used to investigate the reasons for slow I/O performance
- BeeOND: On Demand Filesystem (https://www.beegfs.io/wiki/BeeOND)
  - I/O was done here
- MPIFileUtils: MPI-based tools to handle typical jobs like copy or rsync(https://github.com/hpc/mpifileutils)
  - Used to copy back the data at the end of the job

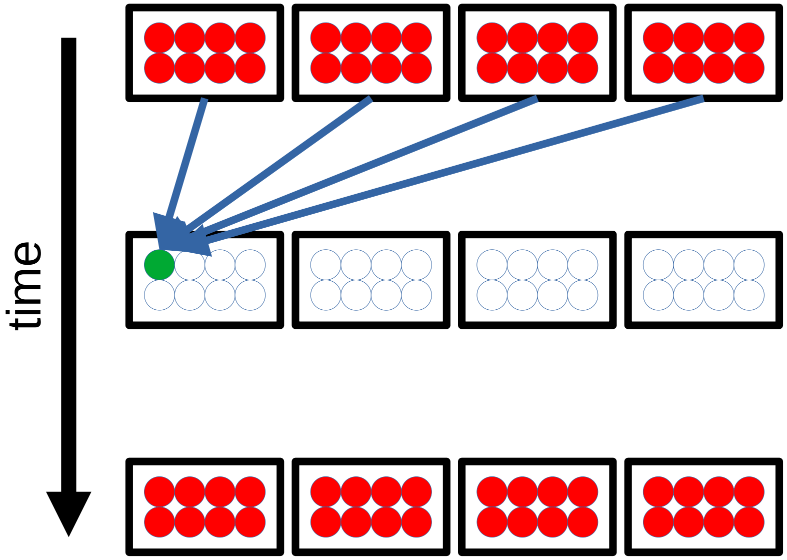
Parallelism in ESMs

- Model area gets decomposed
- Parts of the complete area get distributed on cores
- Each core only sees a small amount of the complete data
- Different strategies for I/O of this data

Disclaimer

- All the following examples are some kind of realistic simulations
- In all cases (realistic) examples exist where I/O influences model performance much more or much less
- If you are doing only little output you don't have to care about this results (for one climate simulation I got I/O was less than 0.1% of total time → no optimizations needed)
- If you are doing a lot of output give it a try

Synchronous serial I/O in ICON and EMAC



All Cores on all nodes do calculations (red)

Data in memory gets transferred to first core

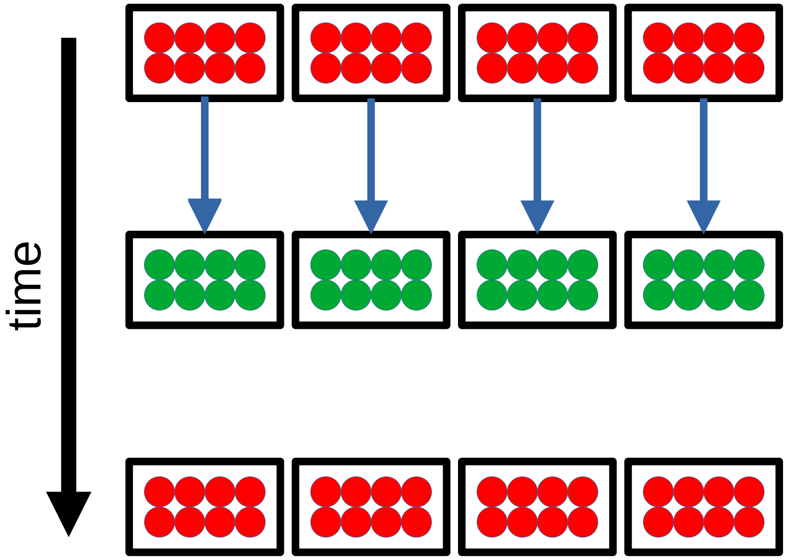
First core (master) does I/O (green), rest of cores idle (white)

When I/O is finished all cores start to work again

Performance of ICON2.5 I/O Benchmark and EMAC

- Difference between I/O on BeeOND and global filesystem within measurement errors
  - Sometimes BeeOND is about 1% faster, sometimes global filesystem is about 1% faster
- Basically the same for EMAC
- Chunks: > 1 MB

Synchronous Parallel I/O in EMAC and OGS



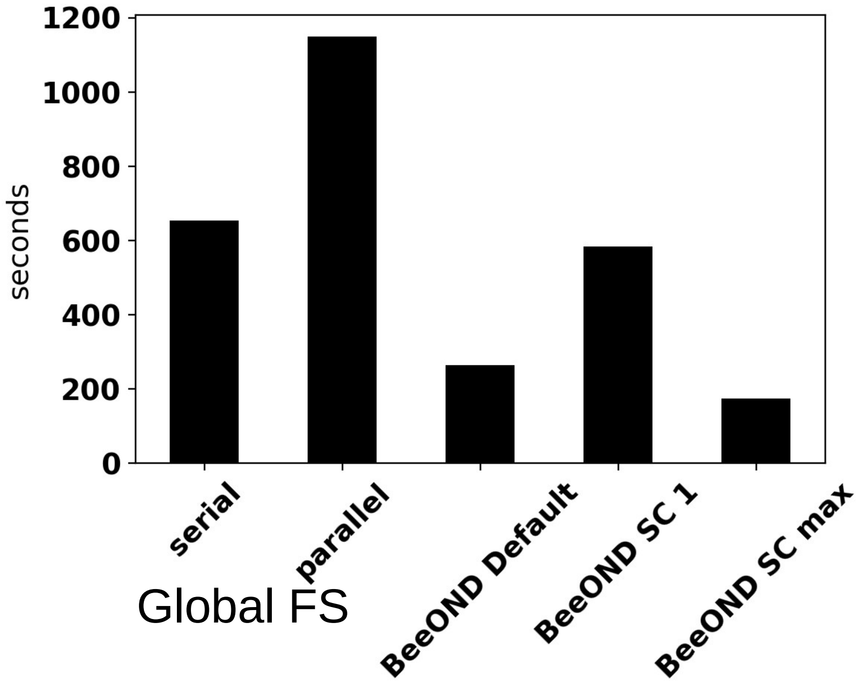
All Cores on all nodes do calculations

Data can remain on cores

All cores do output of their part
- EMAC: all cores into one file
- OGS: each core into a seperate file

When I/O is finished all cores start to work again

EMAC with parallel-netcdf



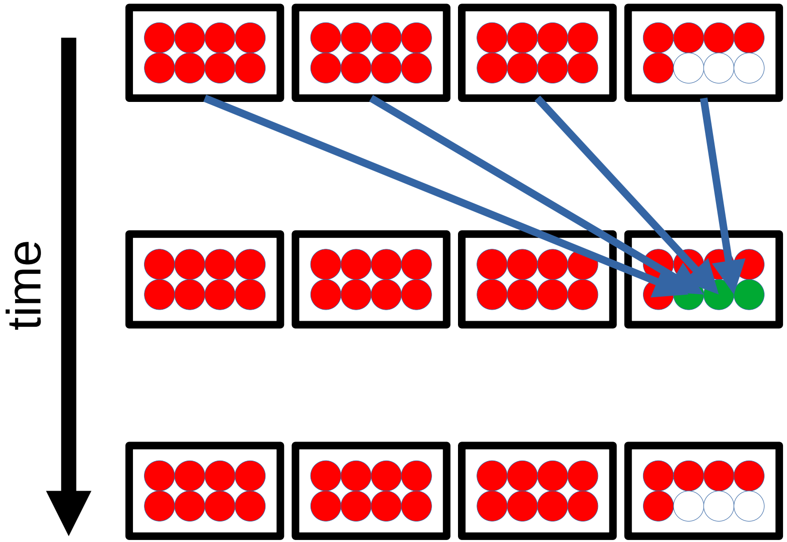
- Example with 20 nodes and high amount of I/O
- Parallel output on global filesystem slower than serial output
- Performance gain on BeeOND for parallel output
  - Stripe count 1: ~factor of 2
  - Default Stripe count 4: ~factor of 4.5
  - Max Stripe count 20: ~factor of 6
- Most common chunk size: a few KB

OGS with pVTU

Time step	Time BeeOND	Time global FS
0	0.62	0.59
1	0.35	0.77
2	0.36	8.73
3	0.35	0.97
4	0.34	0.80
5	0.34	0.75
6	0.37	0.76
7	0.35	10.75
8	0.33	0.87
9	0.34	0.76
10	0.33	0.73

- Example with 480 cores
- With BeeOND about a factor of 2 to 3 faster
- At time step 2 and 7 the global file system was much slower
  - Probably due to interactions by other users on the global file system
- Average chunk size: a few KB

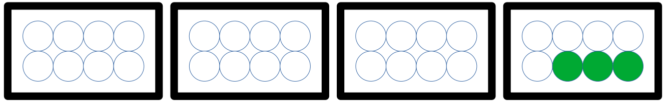
Asynchronous parallel I/O in ICON



Defined number of I/O cores (white) that never take part in calculations

Data in memory gets transferred to I/O cores

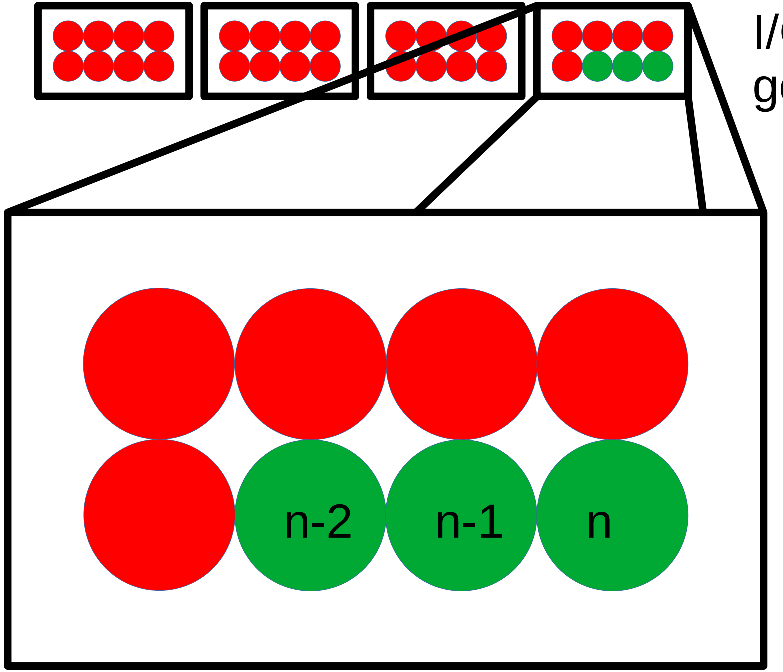
I/O cores do output while all the other cores go on with their work



When I/O is finished before next I/O → I/O cores idle

When I/O is not finished before next I/O → working cores idle

Asynchronous parallel I/O in ICON



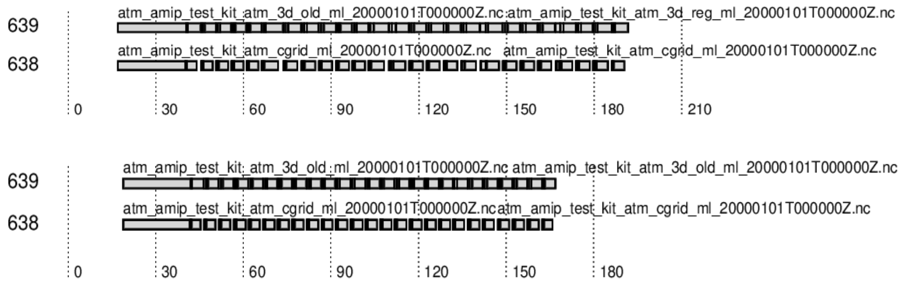
I/O cores do output while all the other cores go on with their work

- Variables are distributed in round robin on I/O cores
- In case of 3 I/O cores
  - Core #n: I/O of file 1, 4, 7, …
  - Core #n-1: I/O of file 2, 5, 8, …
  - Core #n-2: I/O of file 3, 6, 9, …
- Each file can contain a different amount of variables

ICON Version 2.6+: CDI-PIO can be used which can distribute one variable on more than one I/O core (not shown)

ICON 2.6

Output schedule of ICON simulation with 640 nodes and 2 I/O cores



Global FS

BeeOND

Grey bars show when data is written on which core

- Simulation on BeeOND about 20% faster than on global FS
- No effect of BeeOND if there are no waiting times for asynchronous I/O
  - Benefit strongly depends on how you setup I/O in the model simulation

Summary

- I/O can signifcantly slow down ESMs
- Node local on demand filesystems can improve model performance
  - Not so much if large chunks of data are written
  - A lot if very small chunks of data are written

  Side note: Can also be used to directly perform I/O-heavy post processing and further improve overall performance (not shown)