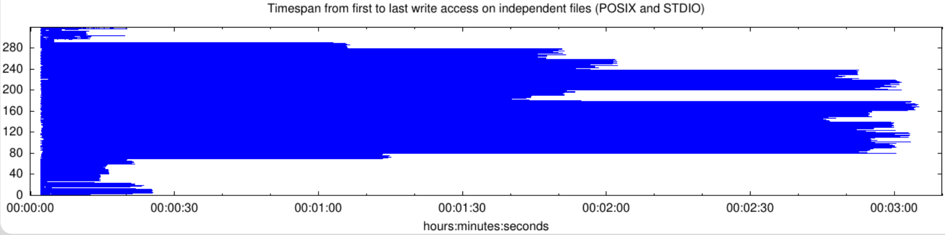
Performance gains in an ESM using parallel ad-hoc file systems

Stefan Versick, Ole Kirner, Jörg Meyer, Holger Obermaier, Mehmet Soysal

Steinbuch Centre for Computing - Data Analytics, Access and Applications



Problem

- 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
  - Example: Large projects like CMIP6 (Climate Model Intercomparison Project) need several Million of CPU-hours and produce data in the Petabyte range per model
- This requires faster high performance computers (HPC), better parallization and **better I/O performance**

Goal

- **Save computing time and diskspace for ESM-simulations**

One way to reach goal

- Helmholtz incubator project Pilot Lab Exascale Earth System Modelling (PL-EESM) working on breakthroughs in ESMs on future exascale computers
- Performance analysis of ESM
- Identify bottlenecks
- Improve I/O performance in ESM
- Create complete workflow for ESM simulations

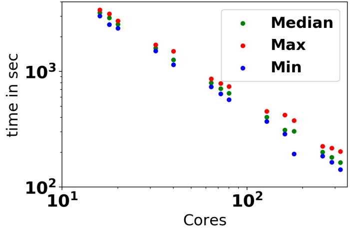
Used Tools

Karlsruhe Institute of Technology

- **EMAC**: Earth System Model; **E**CHAM **M**ESSy **A**tmospheric **C**hemistry; **M**odular **E**arth **S**ubmodel **Sy**stem with ECHAM5 dynamical core (https://www.messy-interface.org/)
- **Scalasca**: Tool for performance analysis of parallelized software (https://www.scalasca.org/)
- **Darshan**: HPC I/O characterization tool (https://www.mcs.anl.gov/research/projects/darshan/)
- **BeeOND**: filesystem BeeGFS On Demand uses local SSDs on compute nodes during the runtime of the job. File system only exists during run time of job and is purged afterwards (https://www.beegfs.io/wiki/BeeOND)
- **MPIFileUtils**: MPI-based tools to handle typical jobs like copy or rsync (https://github.com/hpc/mpifileutils)
- **HeAT**: Python package providing highly optimized algorithms and data structures for tensor computations using CPUs, GPUs and distributed cluster systems on top of MPI (http://www.helmholtz-analytics.de/helmholtz_analytics/EN/GenericMethods/HeAT/_node.html)

Performance analysis of EMAC with the help of Scalasca for single master output

- All analysis done on ForHLR II
- Output with NetCDF library
- Most time in chemistry (good scaling)
- With higher resolution and more output at some point **output is getting dominant because of single master output**



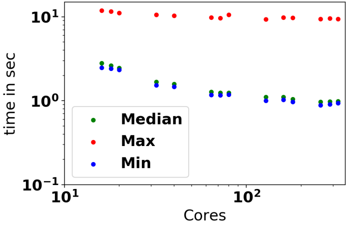
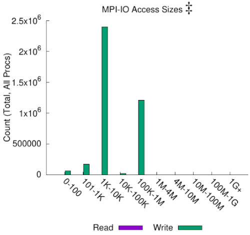


Figure: Min/Median/Max times for cores; left: time spend in physc (mainly chemistry); right: time spend in output; only one core writes (Max), all others only do some communication

Performance analysis of EMAC output with Darshan

- Darshan gives us a detailed look inside what happens in I/O in our model runs
- A lot of **information for each file** written or read
- With the chosen resolution 3D-variables are 320*160*90*8 Bytes (35 MByte) in size, 2D-variables 400 KByte
- With **parallel output** each core writes only a **small chunk** of each variable (slow)
  - In certain circumstances using more cores can slowdown the model



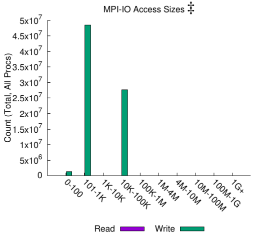
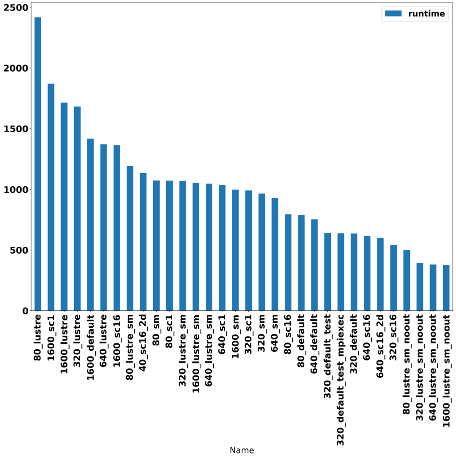


Figure: left: Access sizes using 80 cores; right: Access sizes using 1600 cores

Performance analysis of EMAC output

- EMAC T106 simulation without chemistry and hourly output for one day
- Times include initialization (no parallelism there; about 4 minutes in each run)
- Name scheme: <Number of cores> <file system used; sc: stripe count on BeeOND> <optional: noout: no output>
- In this example: on Lustre runtime is getting smaller from 80 to 640 cores, with 1600 cores runtime getting larger
- In general output done on BeeOND is faster



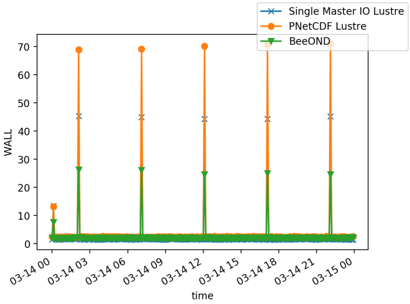
Workflow example

- **Work in progress! Preliminary results**
- **Realistic EMAC model run** including chemistry with T106 resolution (~100 km horizontal resolution)
- We use 80 nodes with 20 cores (in total **1600 cores**)
- In the shown example we are mainly interested in the ozone hole
- We did **four simulations** with different settings
  - **Single Master In- and Output**: One core is doing the whole output on a Lustre file system
  - **PNetCDF Lustre**: Output using parallel-netcdf on a Lustre file system
  - **PNetCDF BeeOND**: Output using parallel-netcdf on a BeeOND file system
  - **PNetCDF BeeOND**: Output using parallel-netcdf on a BeeOND file system with additional post-processing directly on BeeOND
- Each simulation computed 3 model months and produced **8 TByte of output**

Workflow example

Computing times

Model run	Total	Log Overhead	Output	Estimation*
Single Master IO	14.6h	x	5.6h	100%
PNetCDF Lustre	22.3h	x+4.8h	8.7h	119%
PNetCDF BeeOND	15.0h	x+2.6h	3.1h	84%



- Wall-Clock times per timestep
- First small peak: reading data and output for ground stations
- Large Peaks: output timesteps
- Baseline: normal computing plus output for logging data (in a production run this overhead would not be included!)
- * Estimation for time change compared to SMIO model run without logging data

Postprocessing

Karlsruhe Institute of Technology

- Note: Computing Node crashed! All values are **estimations** based on other tests
- Directly at the **BeeOND** filesystem
- In this example we **compute**:
  - Total ozone column
  - Minimum ozone column in high southern latitudes
  - Ozone hole area in high southern latitudes
- **Automatic plots** of the above calculations
- **Packing** a lot of data where high numerical precission is not needed using HeAT; output precission decreased to 16 bit as described in the link
  - **Data reduction** by 90%
- Time needed for postprocessing: 1h (you have to do this anyway)
- Copy back the data using **dcp** from the mpiFileUtils
  - Parallel copying reduces time compared to using just cp
  - **Time saved** compared to non-postprocessed data: 0.2h
- **In future** we additionally will use **lossless compression** for data where high numerical precission is needed

Summary

Karlsruhe Institute of Technology

- ESM simulations need **more and more computation time**
- **Higher parallelization** needed
- One possible **bottleneck: output** of data
- For EMAC **small chunk sizes** when parallel output is used are a **problem** on hard discs with Lustre file systems
- Using **BeeOND** on SSDs **increases performance**
- **Postprocessing** directly on BeeOND can further **improve overall performance**