Utilizing Heterogeneous Storage Infrastructures via the Earth-System Data Middleware

Julian Kunkel on behalf of the ESiWACE WP4 team

Department of Computer Science, University of Reading

26 Sept. 2019

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER
AND CLIMATE IN EUROPE

Outline

Disclaimer: This material reflects only the author's view and the EU-Commission is not responsible for any use that may be made of the information it contains
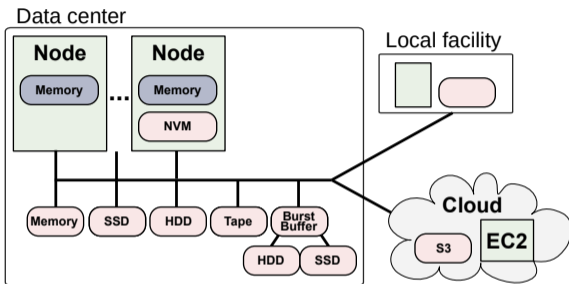
Climate/Weather Workflows

Challenges

- Programming of efficient workflows
- Efficient analysis of data
- Organizing data sets
- Ensuring reproducability of workflows/provenance of data
- Meeting the compute/storage needs in future complex hardware landscape

The Coexistence of Storage – Impact of Local Storage

- ■ Goal: We shall be able to use all storage technologies concurrently
  - ▶ Without explicit migration etc. put data where it fits
  - ▶ Administrators just add new technology (e.g., SSD pool) and users benefit
- ■ May utilize local storage, SSDs, NVMe
  - ▶ Even without communication useful in workflows

Introduction
○○●○

ESDM
○○○○○○○○○○○○○○

Evaluation
○○○○○○○○

Outlook
○○○○○

Summary
○

ESiWACE: http://esiwace.eu

The Centre of Excellence in Simulation of Weather and Climate in Europe

- Prepares the European weather and climate community
  - ▶ to make use of future exascale systems
- Goals in respect to HPC environments:
  - ▶ Improve efficiency and productivity
  - ▶ Supporting the end-to-end workflow of global Earth system modelling
  - ▶ Establish demonstrator simulations that run at the highest affordable resolution
- Funding via the European Union's Horizon 2020 program (ESiWACE2 2019-2022)

The ESiWACE Community

- 20 partners from 9 countries
- 35 supporters

Figure: Group Photo during the ESiWACE2 kickoff (March 2019)

Introduction
0000

ESDM
●000000000000000

Evaluation
00000000

Outlook
00000

Summary
0

Outline

1 Introduction

2 ESDM

3 Evaluation

4 Outlook

5 Summary

Introduction
○○○○

ESDM
○●○○○○○○○○○○○○○

Evaluation
○○○○○○○○

Outlook
○○○○○

Summary
○

Earth-System Data Middleware

A transitional approach towards a vision for I/O addressing

- Scalable data management practice
- The inhomogeneous storage stack
- Suboptimal performance and performance portability
- Data conversion/merging

## Design goals of the Earth-System Data Middleware

1. Relaxed access semantics, tailored to scientific data generation
2. Site-specific (optimized) data layout schemes
3. Ease of use and deployment particularly configuration
4. Enable a configurable namespace based on scientific metadata

Introduction
0000

ESDM
00●00000000000

Evaluation
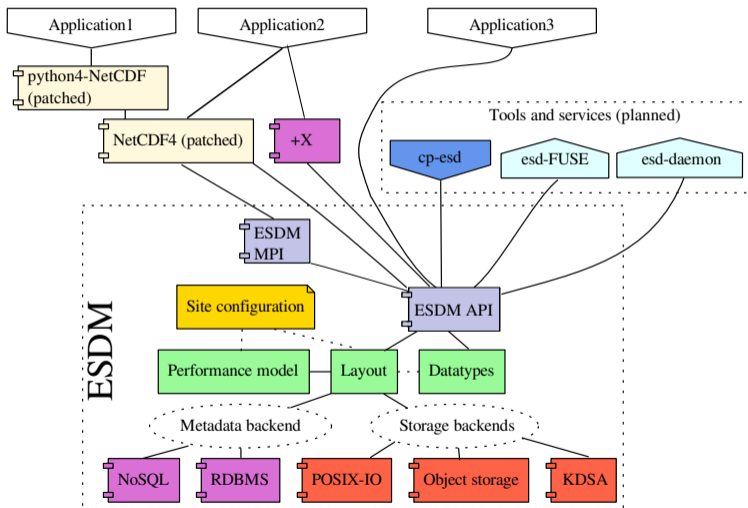00000000

Outlook
00000

Summary
0

Architecture

Key concepts

- Middleware utilizes layout component to make placement decisions
- Applications work through existing API
- Data is then written/read efficiently; potential for optimization inside library

Introduction
○○○○

ESDM
○○○●○○○○○○○○○○

Evaluation
○○○○○○○○

Outlook
○○○○○

Summary
○
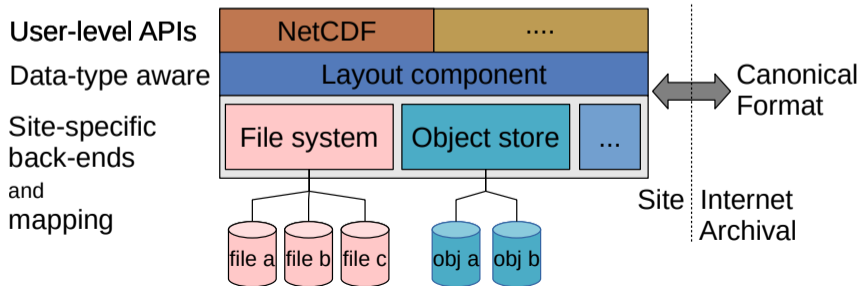
Architecture: Detailed View of the Software Landscape

Data Model

- Container:
  - ▶ Provides a flat (simple hierarchical) namespace
  - ▶ Contains: Datasets + (arbitrary) metadata
  - ▶ Can be constructed on the fly
- Dataset:
  - ▶ Multi-dimensional data of a specified data type
  - ▶ Write-once semantics (epochs are planned)
  - ▶ Contains arbitrary number of data fragments
  - ▶ Data of **different fragments** can be **disjoint or overlapping**
  - ▶ Dimensions can be named and unlimited
  - ▶ Self-describing, can be linked with multiple containers
- Fragment:
  - ▶ Holds data, arbitrary continuous sub-domain (data space)
  - ▶ Stored on exactly one storage backend

Introduction
○○○○

ESDM
○○○○○●○○○○○○○○

Evaluation
○○○○○○○○

Outlook
○○○○○

Summary
○

Discussion of the Data Model

1 Fragment domain is flexible
- ▶ Avoid false sharing (of data blocks) in the write-path
- ▶ A fragment can globally available or just locally
- ▶ Reduce penalties of **shared** file access

2 Self-describing data format
- ▶ Metadata contains relevant scientific metadata, datatypes

3 Layout of the fragments can be dynamically chosen
- ▶ Based on site-configuration and performance model
- ▶ Site-admin/project group defines a mapping
- ▶ Use multiple storages concurrently, use local storage

4 Containers could be created on the fly to mix-in datasets
- ▶ Open one container for input that has everything you need

Backends

Storage backends

- POSIX: backwards compatible for any shared storage
- CLOVIS: Seagate-specific interface, will be open sourced soon
- WOS: DDN-specific interface for object storage
- KDSA: Specific interface for the Kove cluster-wide memory

Metadata backends

- POSIX: backwards compatible for any shared storage
- Investigated performance of ElasticSearch, MongoDB as potential NoSQL solutions

Namespace

- The namespace of ESDM is separated from the file system
- Currently, hierarchically too
- NetCDF can use ESDM by just utilizing the **esdm://** prefix
- Example:

```
$ nccopy test_echam_spectral.nc esdm://user/test_echam_spectral
$ // do something with the file in ESDM, e.g.
$ ncdump -h esdm://user/test_echam_spectral
$ // export the file into the portable NetCDF4 format:
$ $ nccopy -4 esdm://user/test_echam_spectral out.nc
```

Introduction
0000

ESDM
00000000●000000

Evaluation
00000000

Outlook
00000

Summary
0

Metadata of a Complex File: The NetCDF Metadata

```
   netcdf test_echam_spectral {
dimensions:
        time = UNLIMITED ; // (8 currently)
        lat = 96 ;
        lon = 192 ;
        mlev = 47 ;
        ilev = 48 ;
        spc = 2080 ;
        complex = 2 ;
variables:
        float abso4(time, lat, lon) ;
                abso4:long_name = "antropogenic sulfur burden" ;
                abso4:units = "kg/m**2" ;
                abso4:code = 235 ;
                abso4:table = 128 ;
                abso4:grid_type = "gaussian" ;
        ... [126+ more variables] ...
// global attributes:
                :CDI = "Climate Data Interface version 1.4.6 (http://code.zmaw.de/projects/cdi)" ;
                :Conventions = "CF-1.0" ;
                :source = "ECHAM6.1" ;
                :institution = "Max-Planck-Institute for Meteorology" ;
                ... 10 more attributes ...
                :NCO = "4.4.5" ;
}
```

Mapping by the POSIX Metadata Storage

esiwace

Stored metadata inside the metadata directory

```
containers/user/test_echam_spectral.nc.md
datasets/VZ/zMKbbzj9Y0kEpk.md
        ... for each dataset one file ...
```

Metadata is stored as JSON: the container

```
{
  "Variables": { # Metadata of the global attributes
      "childs": {
      "CDI": {
            "data": "Climate Data Interface version 1.4.6 (http://code.zmaw.de/projects/cdi)"
            "type": "q71@l" # The datatype ASCII encoded
        },
      },
  }
  "dsets": [
      {
          "id": "VZzMKbbzj9Y0kEpk",
          "name": "abso4"
      }, ... # for each dataset one ]
}
```

Mapping by the POSIX Metadata Storage

Metadata is stored as JSON: a dataset

```
{ "Variables": {
    "childs": { # Attributes...
    "grid_type": { "data": "gaussian", "type": "q8@l"}
} },
"dims": 3, # dimensionality of the data
"dims_dset_id": [ "time", "lat", "lon"], # the named dimensions
"fill-value": {"data": 9.96920997e+36, "type": "j"},
"size": [0, 96, 192], # the dimensionality of the data, here unlimited 1st dim
"typ": "j" # The type of the data, here float
"id": "VZzMKbbzj9Y0kEpk", # ID of the dataset
"fragments": [
    {"id":"VZzMKbGtnusZsRVv3Pky","pid":"p1","size":[1,96,192],"offset":[0,0,0]},
    {"id":"VZzMKbRhYpI6cOl0frBX","pid":"p1","size":[1,96,192],"offset":[1,0,0]},
    ...
    {"id":"VZzMKbl8JyXk4fUXfwrS","pid":"p1","size":[1,96,192],"offset":[7,0,0]}]
}
```

Introduction
0000

ESDM
00000000000●00

Evaluation
00000000

Outlook
00000

Summary
0

Mapping of Fragments by Storage Backends

Mapping of the POSIX storage

- A fragment is mapped into a file: `<dataset>/<fragmentID>`
- Contains the raw data
- Optionally suffixed by some metadata to allow "restoration" of broken storage

Mapping of the KDSA storage

- Volume of shared memory is partitioned into blocks
- Block header describes free/occupied blocks
- Atomic operations to aquire/free a block
- A block stores one fragment, ID is the offset into the volume

Introduction
0000

ESDM
00000000000000●0

Evaluation
00000000

Outlook
00000

Summary
0

The Blocking I/O Path: `Write`

- Note: a processes write path is independent from any global state

1 Scheduler identifies how to partition the data into fragments and assigns backends
  - ▶ A maximum fragment size is defined by each backend
  - ▶ May also use a performance model to partition data
  - ▶ (We aim to utilize workflow information for the partitioning)

2 Append the fragment to the local dataset (mark as dirty)

3 A backend-specific thread pool processes the fragments
  - ▶ The backend is called with the fragment
  - ▶ May use direct-I/O or reorganize the data in-memory

4 Wait until all fragments are processed

Collective operation

5 Upon close/sync, the MPI interface synchronizes the fragment knowledge

6 A single process updates the JSON metadata for the dataset/container

Introduction
0000

ESDM
000000000000●

Evaluation
00000000

Outlook
00000

Summary
0

The Blocking I/O Path: Read

Preliminaries – Collective open/ref. operation of a dataset/container

1 Upon open, the fragment information is read by one process

2 Broadcast fragment information to all processes

3 Identify the overlap of fragments with the data space requested

4 Make a schedule to read each cell once (there could be replicas)

5 A backend-specific thread pool processes the fragments
  ▶ Backend loads the fragments requested (use direct-I/O or copy data if needed)

6 Wait until all fragments are processed

Introduction
0000

ESDM
00000000000000

Evaluation
●0000000

Outlook
00000

Summary
0

Outline

1 Introduction

2 ESDM

3 Evaluation

4 Outlook

5 Summary

Introduction
0000

ESDM
0000000000000

Evaluation
0●000000

Outlook
00000

Summary
0

Evaluation

System

- Test system: DKRZ Mistral supercomputer
- Nodes: 100, 200, 500

Benchmark

- Uses ESDM interface directly; Metadata on Lustre
- `Write`/`read` a timeseries of a 2D variable; 3x repeated
- Grid size: 200k $\times$ 200k $\times$ 8 Bytes $\times$ 10 iterations
- Data volume: size = 2980 GiB; compared to IOR performance

ESDM configurations

- Splitting data into fragments of 100 MiB
- Use /dev/shm (TMPFS) or /tmp directory (Local SSD)

Introduction
0000

ESDM
0000000000000

Evaluation
00●00000

Outlook
00000

Summary
0

Performance Growth of ESDM on Lustre (PPN = 1)
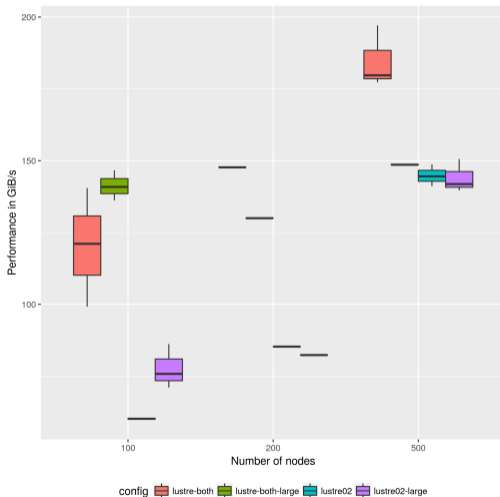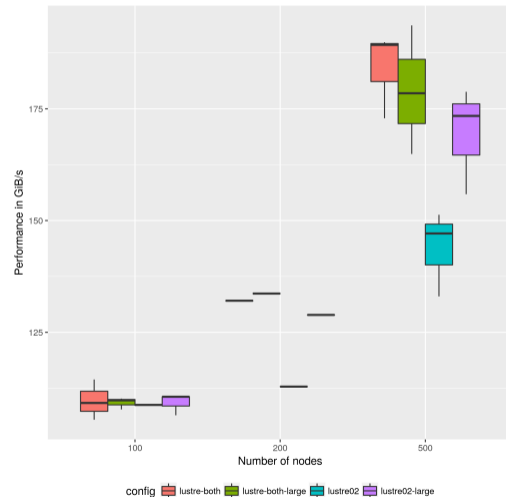

Figure: `Write`

Figure: `Read`

Discussion

- Benefit when accessing multiple global file systems
- Write performance benefits from using both file systems
  - ▶ Most benefit when using 200 nodes (2x)
  - ▶ 500 nodes: 180 GiB/s vs. 140 GiB/s (single fs)
- Read performance shows some benefit for larger configurations
- ESDM achieves similar performance regardless of PPN (not shown)
- What is the performance when we use node-local storage?
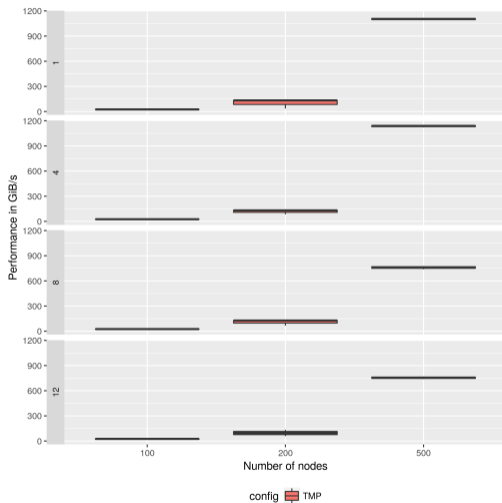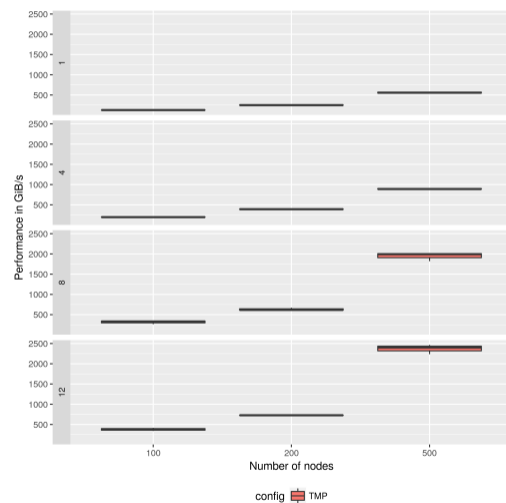
ESDM on TMP (varied PPN and nodes)

Figure: `Write`

Figure: `Read`

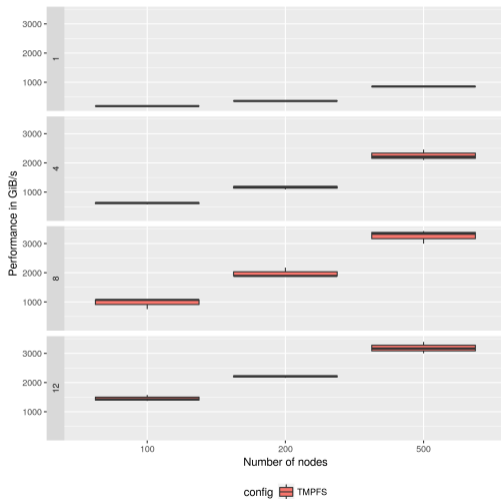# ESDM on TMPFS – Providing Potential (varied PPN and nodes)

Figure: Write

Figure: Read

Introduction
0000

ESDM
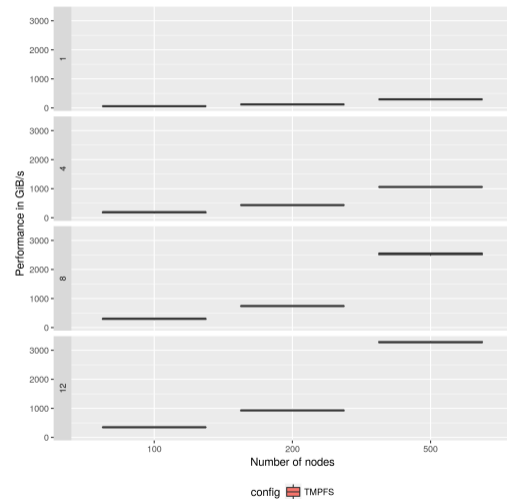0000000000000

Evaluation
00000000

Outlook
00000

Summary
0

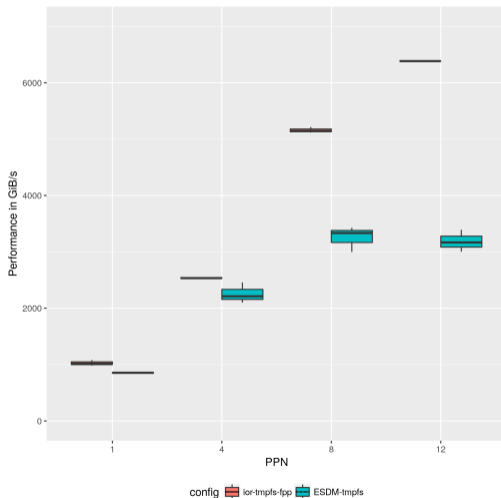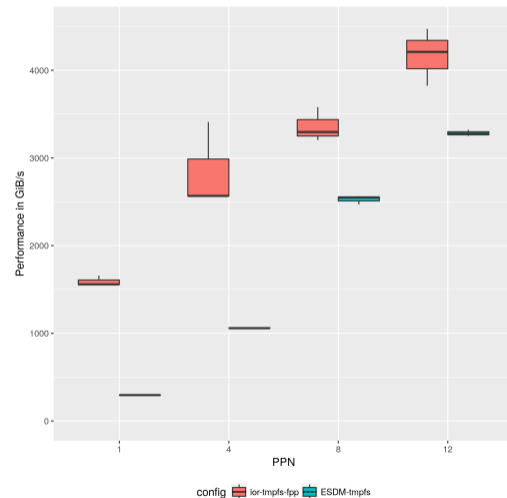# Performance on TMPFS vs IOR (nodes=500, varied PPN)

Figure: `Write`

Figure: `Read`

Discussion

- Node-local storage is much faster than global storage
  - ► TMP achieves 750-1,000 GB/s for write (500 SSDs, some caching)
  - ► TMP reads are actually cached (6 GB data per node)
  - ► TMPFS achieves up to 3,000 GB/s
- TMP write is invariant to PPN
  - ► Rason: configured ESDM to use at least 4 threads per node
- TMPFS write depends on PPN
  - ► ESDM configured to not use threads, could use them to improve performance!
- IOR is faster: potential to improve ESDM path further
  - ► Localization of fragments using r-tree

Introduction
0000

ESDM
0000000000000

Evaluation
00000000

Outlook
●0000

Summary
0

Outline

1 Introduction

2 ESDM

3 Evaluation

4 Outlook

5 Summary

Status

- NetCDF: done, minor issues to fix, use tests for checking compatibility
  - ▶ netcdf4-python: available, derived tests with supported features
  - ▶ Report for compatibility will appear soon (Oct. 2019)
  - ▶ Some unsupported features, e.g., NetCDF4-groups, will be done depending on needs
- First tools implemented (esdm-mkfs, esdm-rm)
- Deployed daily regression testing using Jenkins (Webpage to go public: Oct. 2019)
- FUSE prototype to dynamically build a hierarchical namespace on semantics
  - ▶ e.g., <model>/<date>/<variable>

Introduction
oooo

ESDM
oooooooooooooo

Evaluation
ooooooooo

Outlook
ooo●o

Summary
o

ESiWACE2 Plans for ESDM

- Hardening and optimization of ESDM
  - Performance optimization of the `read` path (fragments involved in I/O)
  - Replicate data upon `read`
- Integrate an improved performance model
- Industry proof of concepts for EDSM, i.e., shipping of HW with software
- Improvements on data compression (also for NetCDF)
- Optimized backends for, e.g., Clovis, IME, S3
- Workflow support and active storage

Introduction
oooo

ESDM
oooooooooooooo

Evaluation
ooooooooo

Outlook
ooooo●o

Summary
o

ESiWACE2 Plans for ESDM: Workflows and Active Storage

- Integrate Cylc and ESDM (requires additional API, the +X on Slide 11)
  - ► Extensions to Cylc to cover data lifecycle, e.g., I/O performance needs
  - ► Cylc to provide information about workflow to ESDM/scheduler
  - ► ESDM to make better placement decisions and better use `node-local storage`
- Supporting post-processing, analytics and (in-situ) visualization
  - ► Support of computation offloading within ESDM
  - ► Integration with analysis tools, e.g., Ophidia, CDO
  - ► Sending fragment data directly to another process

Offloading of computation (part of the +X on Slide 11)

- Consider a reduction operation such as `max()` and more complex ops
- The storage could perform this operation on data of each fragment (independently)
  - ► Clovis already offers such an API
  - ► Working together with DDN to provide it in IME

Introduction
0000

ESDM
0000000000000

Evaluation
00000000

Outlook
0000●

Summary
0

Long Term Vision: Full Separation of Concerns

Decisions made by scientists

- Scientific metadata
- Declaring workflows
  - ▶ Covering data ingestion, processing, product generation and analysis
  - ▶ Data life cycle (and archive/exchange file format)
  - ▶ Constraints on: accessibility (permissions), ...
  - ▶ Expectations: completion time (interactive feedback human/system)
- Modifying workflows on the fly
- Interactive analysis, e.g., Visual Analytics
- Declaring value of data (logfile, data-product, observation)

Summary

Software

1. ESDM: Performance-portable I/O utilizing heterogeneous storage
2. The data model is mostly backwards compatible to NetCDF
3. NetCDF/Python workflows supported
4. Working towards workflow and active storage support
5. Ongoing: exploiting `node-local storage` better