2020 Summer School on Effective HPC for Climate and Weather

Input/Output and Middleware

Luciana Pedro, Julian Kunkel

Department of Computer Science, University of Reading

18 June 2020

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

Learning Objectives – Talk

- Discuss challenges for data-driven research (Section Introduction)
- Describe the role of middleware and file formats (Section I/O Solutions)
- Identify typical I/O performance issues and their causes (Section I/O Solutions)
- Apply performance models to assess and optimize the application I/O performance (Section I/O Performance)
- Design a data model for NetCDF/CF (Section NetCDF)
- Implement an application that utilizes parallel I/O to store and analyze data (Section Parallel I/O)
- Describe ongoing research activities in high-performance storage (Section Research Activities)

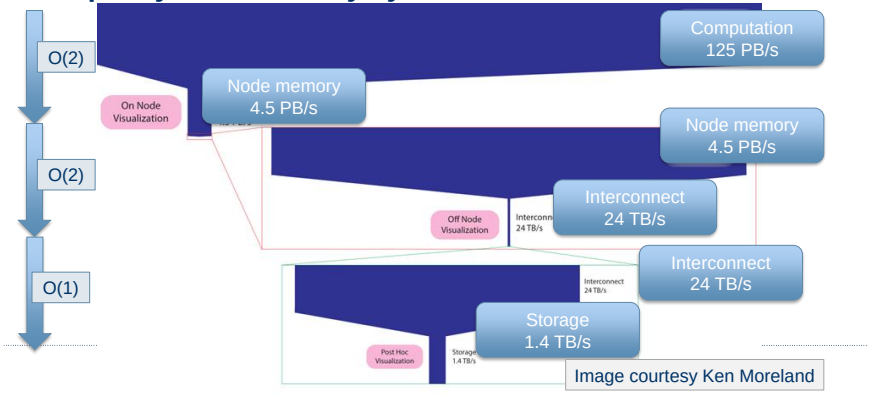
Learning Objectives – Lab

- Execute programs in C that read and write NetCDF files in a metadata-aware manner
- Analyze, manipulate and visualise NetCDF data
- Implement an application that utilizes parallel I/O to store and analyze data

Outline

I/O Bottleneck – Titan

Five orders of magnitude between compute and I/O capacity on Titan Cray system at ORNL



I/O Bottleneck – Mistral

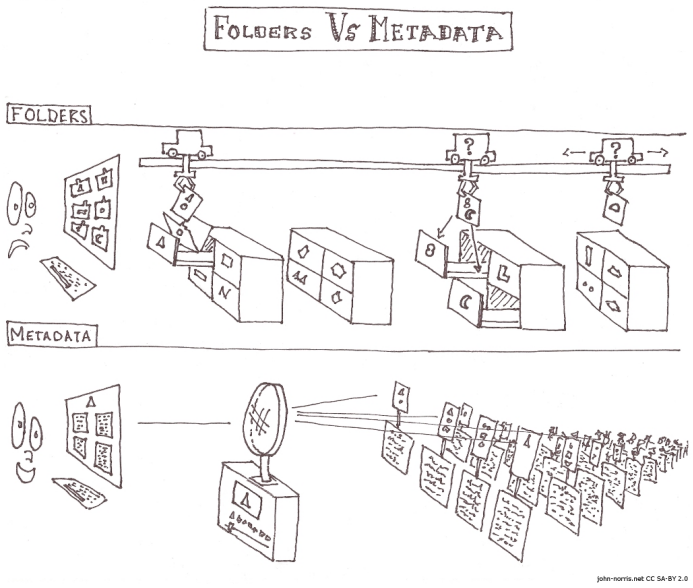
- Predict processor performance growth by 20x each generation (∼5 years).
- Storage throughput/capacity improves by just 6x.
- Costs and performance come together.

Exascale Storage Systems – An Analytical Study of Expenses

	2004	2009	2015	2020	2025	Exascale (2020)
Performance	1.5 TF/s	150 TF/s	3 PF/s	60 PF/s	1.2 EF/s	1 EF/s
Nodes	24	264	2500	12,500	31,250	100k-1M
Node performance	62.5 GF/s	0.6 TF/s	1.2 TF/s	4.8 TF/s	38.4 TF/s	1-15 TF/s
System memory	1.5 TB	20 TB	170 TB	1.5 PB	12.8 PB	3.6-300 PB
Storage capacity	100 TB	5.6 PB	45 PB	270 PB	1.6 EB	0.15-18 EB
Storage throughput	5 GB/s	30 GB/s	400 GB/s	2.5 TB/s	15 TB/s	20-300 TB/s
Disk drives	4000	7200	8500	10000	12000	100k-1000k
Archive capacity	6 PB	53 PB	335 PB	1.3 EB	5.4 EB	7.2-600 EB
Archive throughput	1 GB/s	9.6 GB/s	21 GB/s	57 GB/s	128 GB/s	-
Power consumption	250 kW	1.6 MW	1.4 MW	1.4 MW	1.4 MW	20-70 MW
Investment	26 M€	30 M€	30 M€	30 M€	30 M€	$200 M[4]

Real Values – 2018

	Mistral
Characteristics	**Value**
Performance	3.1 PF/s
Nodes	2882
Node performance	1.0 TF/s
System memory	200 TB
Storage capacity	52 PB
Storage throughput	700 GB/s
Disk drives	10600
Archive capacity	500 PB
Archive throughput	18 GB/s
Compute costs	15.75 M EUR
Network costs	5.25 M EUR
Storage costs	7.5 M EUR
Archive costs	5 M EUR
Building costs	5 M EUR
Investment	38.5 M EUR
Compute power	1100 kW
Network power	50 kW
Storage power	250 kW
Archive power	25 kW
Power consumption	1.20 MW


FOLDERS VS METADATA
FOLDERS
METADATA
john-norris.net CC SA-BY 2.0

Data-driven Research

- **Data-driven Research** is the science of letting data tell us what we are looking for.
  - **Database Management** is the science of efficiently storing and retrieving data.
  - **Data Mining** is the science of discovering hidden correlations in data.
- In HPC, the concerns of **storage** and **computing** are traditionally separated and optimised independently from each other and the needs of the end-to-end user.
- Workflows composed of data, computing, and communication-intensive tasks should drive interfaces and hardware configurations to best support the programming models.
- Data-driven workflows may benefit from the explicit and simultaneous use of a locally heterogeneous set of computing and storage technologies.

Outline

Input/Output

- Input/Output (I/O) is simply data migration.
  - Memory $\Leftrightarrow$ Disk
- I/O is a very expensive operation!
- How is I/O performed?
  - I/O Pattern
    - Number of processes and files.
    - Characteristics of file access.
- Where is I/O performed?
  - Characteristics of the computational system.
  - Characteristics of the file system.

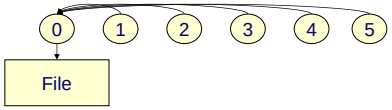
I/O Performance

- There is no "One Size Fits All" solution to the I/O problem.
- Bottlenecks in performance can occur in many locations.
  - Application and/or file system.
- Many I/O patterns work well for some range of parameters.
- Going to extremes with an I/O pattern will typically lead to problems.
- **Golden Rule:** Increase performance by decreasing the number of I/O operations and increasing the size of each operation.

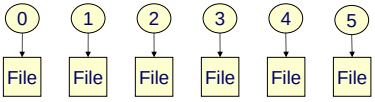
I/O Types

Serial, multi-file parallel and shared file parallel I/O

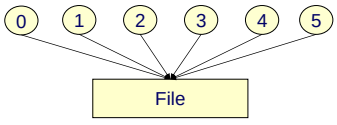
Serial I/O



Parallel Multi-file I/O

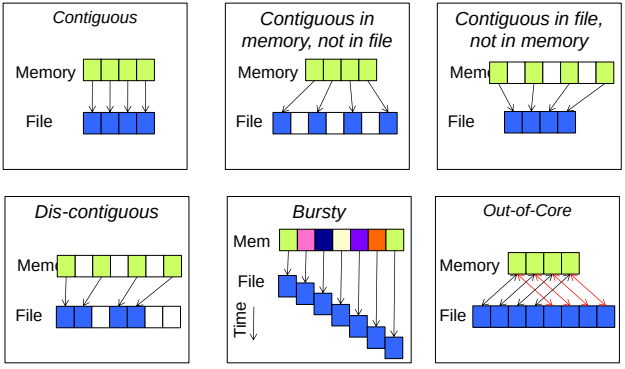


Parallel Shared-file I/O



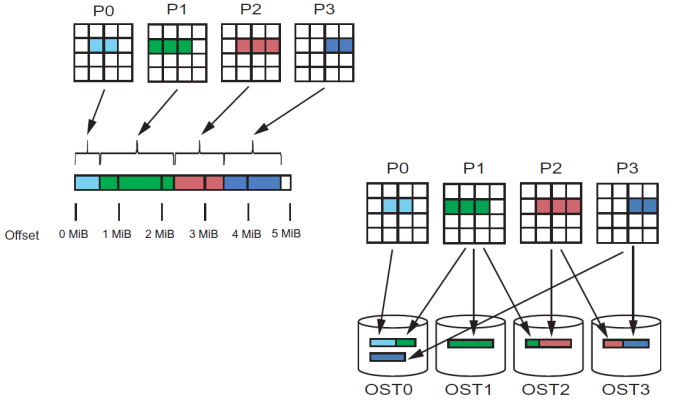
I/O Access Patterns

Access Patterns



File Striping

File Striping: Physical and Logical Views



16 ©2009 Cray Inc.

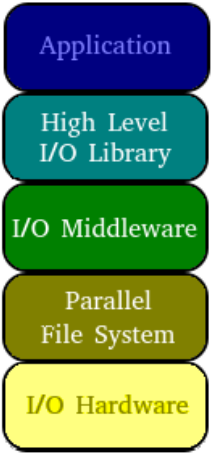
I/O Stack

Application

- Weather forecasts.
- Climate simulations, impacts, predictions and projections.
- Decisions on emission reductions.
- Strategies for housing, cities, farming, coastal defenses and other parts of society.

High Level I/O Library

- Match storage abstraction to domain.
- Provide self-describing, structured files.
- Map to middleware interface.
- Implement further optimizations.



I/O Middleware

- Match the programming model (e.g. MPI).
- Facilitate concurrent access by groups of processes.
- Expose a generic interface.
- Efficiently map middleware operations into operations in the Parallel File System.

Parallel File System

- Manage storage hardware.
- In the I/O software stack, focus on concurrent, independent access.
- Publish an interface that middleware can use effectively.

I/O Problems

- Not enough I/O capacity on current HPC systems, and the trend is getting worse.
- If there is not enough I/O, you cannot write data to storage, so you can not analyze it.
  - Lost science!
- Missing opportunity of doing better science (analysis) when have access to full spatiotemporal resolution data.
- Energy consumption: it costs a lot of power to write data to disk.

Challenges in Domain of Climate and Weather

- High data volume and velocity
- Data management practice does not scale
  - E.g., hierarchical namespaces does not reflect use cases
  - Scientists spend quite some time to define the namespace
- Suboptimal performance (and performance portability) of data formats
  - Tuning for NetCDF, HDF5 and GRIB necessary
  - Scientists worry about interoperability
- Data conversion is often needed
  - Between formats such as NetCDF and GRIB
  - To combine data from multiple experiments, time steps, etc.
- External data services to share data in the community
  - (Scientific) metadata is provided by databases

Outline

I/O Solutions

- As we are moving towards exascale, the gap between computing power and I/O bandwidth will widen and researchers are looking for solutions to tackle this problem.
- There are essentially three lines of research:
  - at hardware level,
  - at middleware level,
  - and at application level.

Hardware Level

- Non-volatile memory (NVM)
  - Non-volatile memory (NVM) is a type of computer memory that can retrieve stored information even after having been power cycled.
  - The capabilities of NVM (i.e., capacity, bandwidth, energy consumption) are somewhere in-between main memory and persistent storage, thus it is often used as a "caching" solution between these two layers.
  - Examples of non-volatile memory include flash memory, read-only memory (ROM), ferroelectric RAM, most types of magnetic computer storage devices (e.g. hard disk drives, floppy disks, and magnetic tape), optical discs, and early computer storage methods such as paper tape and punched cards.

Hardware Level

- Burst buffer (BB)
  - Burst buffer (BB) is a fast and intermediate storage layer positioned between the front-end computing processes and the backend storage systems.
  - HPC applications often show bursty I/O behavior (i.e., all processes read/write at the same time) and burst buffers help to absorb these workloads.
  - Burst buffer is built from arrays of high-performance storage devices, such as NVRAM and SSD.

Middleware Level

- Solutions in I/O middleware.
  - E.g., file systems, I/O interfaces.
- **Damaris:** Software framework that overlaps computation and I/O operations by dedicating a single core to I/O tasks.
- **ADIOS:** I/O abstraction framework for HPC applications that enables switching between different I/O transport methods with little modification to application code and enabling integration of new I/O solutions.
- **DeltaFS:** File systems that improves the scalability of file systems by letting compute nodes manage metadata instead of a centralized server.

Ongoing Activity: Earth-System Data Middleware

- ESDM provides 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
- Part of the ESiWACE Project[1]

[1] Centre of Excellence in Simulation of Weather and Climate in Europe.

Application Level

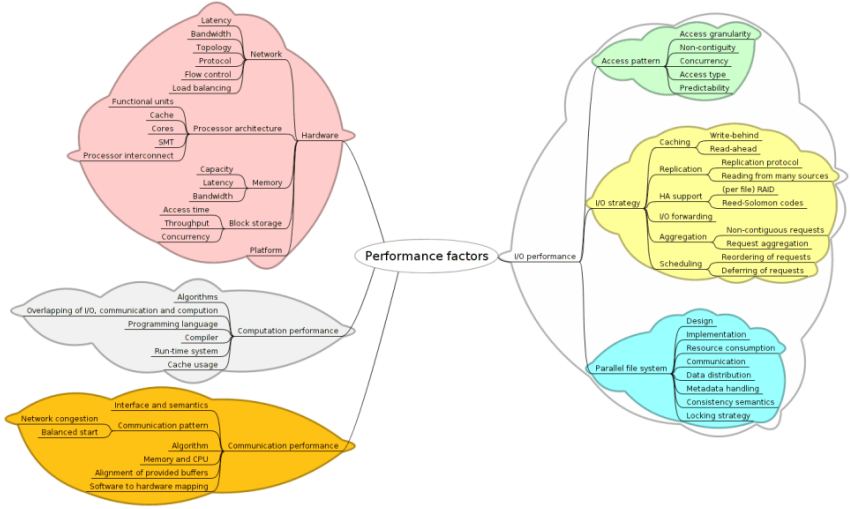
- In-situ analysis
  - In biology and biomedical engineering, in situ means to examine the phenomenon exactly in place where it occurs (i.e., without moving it to some special medium).
  - Rather than applications writing their raw output to storage to later be read again for post-processing (e.g., visualization, filtering, statistics), in-situ processing removes this overhead by performing the analysis directly on the same machines as where the applications run.
  - ParaView, Dax, and Damaris/Viz are tools for large-scale in-situ visualization.

Discussion

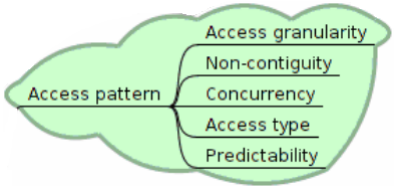
- No "One Size Fits All" solution to the storage problem and programmers must take I/O into careful consideration when developing applications.
- Mismatch between the massive computational performance of processors and relatively limited I/O bandwidth of storage systems.
- Three methods to alleviate this problem: new hardware technology, new I/O middleware, and application-specific solutions.
  - Hardware technology shows promising solutions, but different systems might employ different solutions, reducing the portability and increasing the complexity.
  - Middleware can alleviate some of this complexity with solutions such as ADIOS and ESDM.
  - In-situ analysis is an example of how application-specific solutions can be used to improve I/O throughput and thus application performance.

Outline

Typical Performance Factors



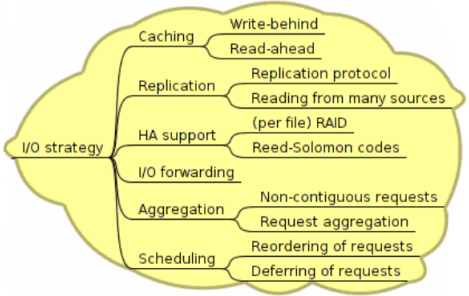
I/O Performance Factor – Access Patterns



- The access pattern describes how spatial access is performed over time.
- With an access pattern, the I/O of a single client process can be described, but also the actual observable patterns on the I/O servers, or on a single block device.
- The pattern on the I/O servers is caused by all clients and defines the performance of the I/O subsystems.

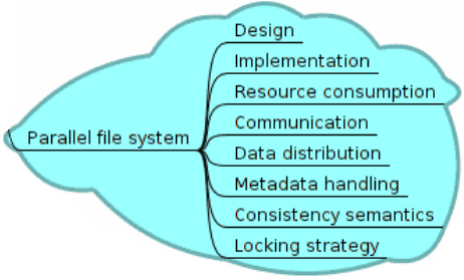
I/O Performance Factor – I/O Strategy

- In general, the mechanisms introduced here are orthogonal to the hardware and the architecture of the parallel file system.
- On the client-side, for instance, requests could already be tuned to improve the access pattern which will be observed on the servers.
- Similar to optimizations in communication, these strategies could be applied on any layer involved in I/O.



I/O Performance Factor – Parallel File System

- Performance of a parallel file system highly depends on its design as it provides the frame for the deployed optimization strategies.
- Several aspects like consistency semantics also apply to higher level interfaces like domain specific I/O libraries.

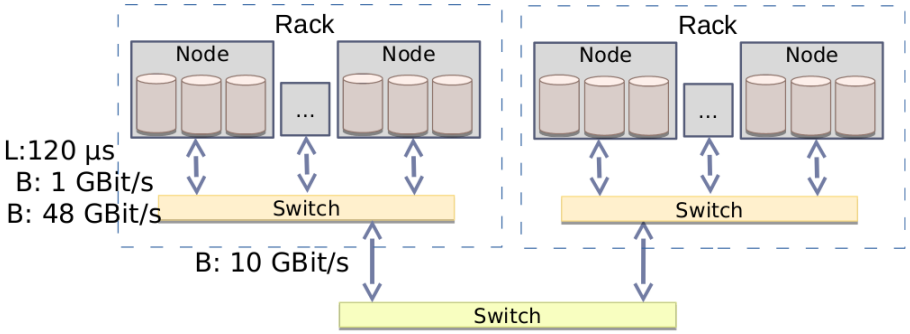


I/O Performance Analysis

- Problem
  - Assessing observed time for I/O is difficult.
  - What best-case performance can we expect?
- Support for performance analysis
  - Models and simulation
    - Trivial models: using throughput + latency
    - PIOSimHD: MPI application + storage system simulator
  - Tools to capture and analyze system statistics and I/O activities
    - HDTrace – Tracing tool for parallel I/O (+ PVFS2)
    - SIOX – Tool to capture I/O on various levels
    - Grafana – Online monitoring for DKRZ (support)
  - Benchmarks – On various levels, e.g., Metadata (md-workbench, IOR)
  - Statistic model to determine likely cause based on time

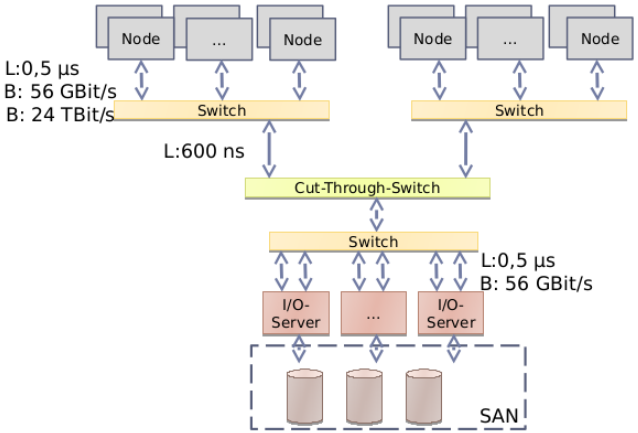
Big Data Cluster Characteristics

- Usually commodity components
- Cheap (on-board) interconnect, node-local storage
- Communication (bisection) bandwidth between different racks is low



HPC Cluster Characteristics

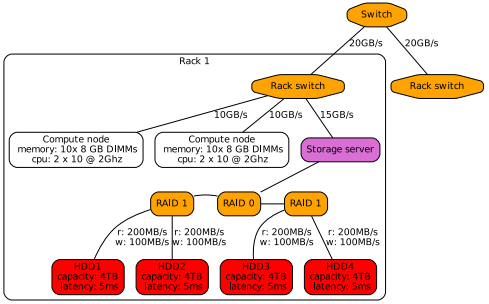
- High-end components
- Extra fast interconnect, global/shared storage with dedicated servers
- Network provides high (near-full) bisection bandwidth.



Performance: Hierarchical Model LR: Extra slide

Example: Performance

Hierarchical model including different levels of detail.



Hardware Performance

Computation

- CPU performance (frequency $\times$ cores $\times$ sockets)
  - E.g.: 2.5 GHz $\times$ 12 cores $\times$ 2 sockets = 60 Gcycles/s
  - The number of cycles per operation depend on the instruction stream
- Memory (throughput $\times$ channels)
  - E.g.: 25.6 GB/s per DDR4 DIMM $\times$ 3

Communication via the network

- Throughput, e.g., 125 MiB/s with Gigabit Ethernet
- Latency, e.g., 0.1 ms with Gigabit Ethernet

I/O Performance Analysis – A Simple Model

- Communication between different machines is limited by the network performance.
- **Network throughput** is the amount of data moved successfully from one place to another in a given time period. The maximum throughput depends on the number of storage servers, client nodes and their network connectivity.
- Users typically know:
  - Output/input file size.
  - Number of nodes a job run on.
  - Runtime of the I/O during a job (this can be obtained with simple means).
- Now compute the observed throughput (`tp_obs`) in MiB/s per node. If `tp_obs` is much smaller than the network throughput, then there might be an I/O problem.
- This is a very basic model that every user should understand and apply.

I/O Performance Analysis – Numeric Example

- Consider the following scenario:
  - File size: 10 GiB
  - Number of nodes: 10
  - Time to transfer the data: 10 seconds
- Calculating the throughput in this example, one will find:
  - $$\frac{10\ \text{GiB}}{10\ \text{nodes} \times 10\ \text{seconds}} = 0.1\ \text{GiB/s per node} = 100\ \text{MiB/s per node}.$$
- Considering that a Gigabit Ethernet network should be capable of delivering a theoretical maximum transfer of about 125 MB/s LR: MiB/s??, the estimate throughput is close to the optimal value.
- However, if you are using an Infiniband capable of delivering 6 GiB/s, then 0.1 GiB/s is a problem. LR: I couldn't verify 6 GiB/s. There are lots of options! So I wrote as an example.

I/O Performance Analysis – Latency

- **Network latency** is the time between the sending of a message and its arrival at the receiver side.
- **Network bandwidth** is the number of bits which can be transferred in a specific time.
- Because protocols like TCP have some overhead and control algorithms, the throughput is smaller than the bandwidth.
- Latency and bandwidth depend on the used network technology and topology.
- Say, for instance, you also know the number of I/O calls as well. Then, you can compute the latency per call.
  - This information can actually be measured using Darshan, for example.
- Now compute the calls per second per node and relate it to the network latency.

Improving I/O Performance

- Software and hardware tries to hide I/O penalty.
- Caching of data:
  - Allow application to continue while I/O completes in the background (write-behind).
  - Allow to aggregate multiple (small) operations into larger operations.
  - Read data from disk before it is needed (read-ahead).
  - **Require memory! Hiding vs. increased problem size!**.
- Programming:
  - Overlap I/O (or communication) with computation.
  - I/O and communication comes almost for free.
  - Optimize file format and access pattern (more complicated).

Outline

NetCDF

- In a simple view, NetCDF is:
  - A data model.
  - A file format.
  - A set of APIs and libraries for various programming languages.
- Together, the data model, file format, and APIs support the creation, access, and sharing of scientific data.
- NetCDF allows the user to describe multidimensional data and include metadata which further characterizes the data.
- NetCDF APIs are available for most programming languages used in geosciences.

Common Data form Language (CDL)

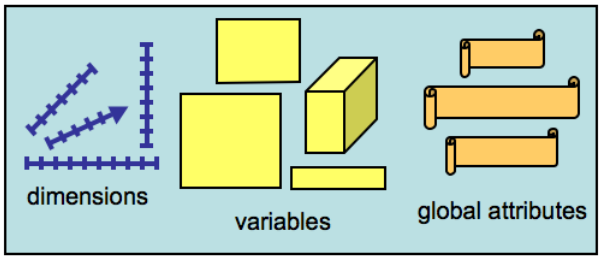
- The notation used to describe a NetCDF object is called CDL (network Common Data form Language), which provides a convenient way of describing NetCDF datasets.

```
netcdf short {
dimensions:
  latitude = 3 ;
  longitude = 2 ;
variables:
  float sfc_temp(latitude, longitude) ;
    sfc_temp:units = "celsius" ;
data:

 sfc_temp =
  10, 10.1,
  10.2, 10.3,
  10.4, 10.5 ;
}
```

- The NetCDF system includes utilities for producing human-oriented CDL text files from binary NetCDF datasets and vice-versa.

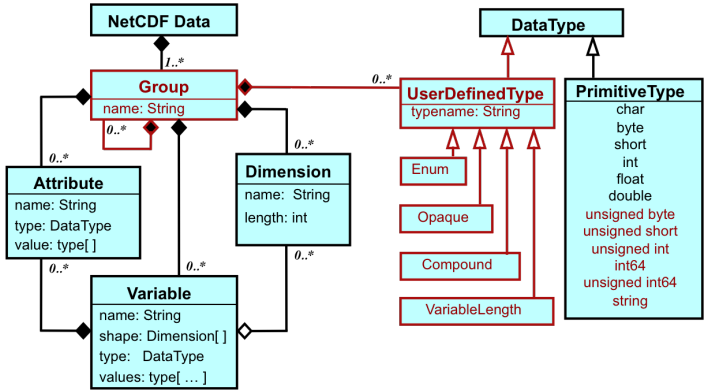
The Classic NetCDF Model

- A NetCDF file (dataset) has a path name and possibly some dimensions, variables, global (file-level) attributes, and data values associated with the variables.



NetCDF Data Models

- Classic: Simplest model – Dimensions, variables, attributes
- Enhanced: More powerful model – Adds groups, types, nesting



The NetCDF-4 Enhanced Data Model

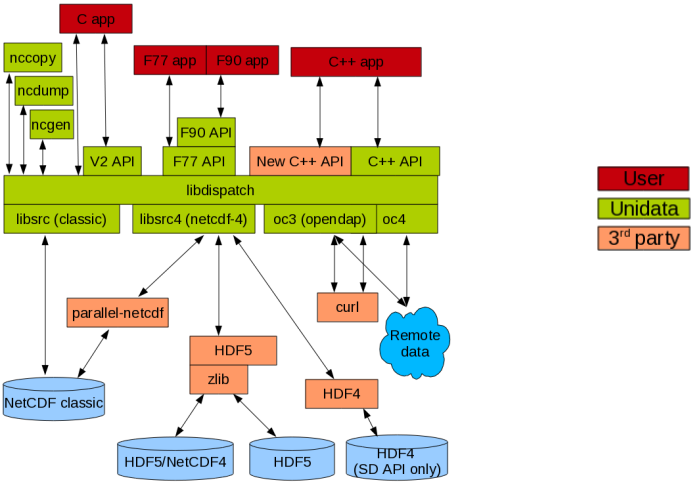
- The NetCDF-4 Enhanced Data Model, which is known as the "Common Data Model", is part of an effort of Unidata to find a common engineering language for the development of scientific data solutions.

- The model contains the variables, dimensions, and attributes of the classic data model, but adds:
  - Groups – A way of hierarchically organizing data, similar to directories in a Unix file system.
  - User-defined types – The user can now define compound types (like C structures), enumeration types, variable length arrays, and opaque types.

The NetCDF-4 Enhanced Data Model

- A file has a top-level unnamed group.
- Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes.
- Variables also have attributes.
- Variables may share dimensions, indicating a common grid.
- One or more dimensions may be of unlimited length.

NetCDF Library Architecture



Experience-based "Best Practices" for Writing NetCDF Files

- Conventions
  - Developers should be familiar with and use existing NetCDF conventions.
- Coordinate Systems
  - Spatial and temporal location of data are supported by use of coordinate systems.
- Variable Grouping
  - How you group data into variables can determine whether common analysis and visualization software can effectively use the data.
- Variable Attributes
  - Conventional variable attributes supply necessary metadata.

Experience-based "Best Practices" for Writing NetCDF Files

- Strings and Character Variables
  - Use character data properly for representing text strings.
- Calendar Date and Time
  - Represent calendar dates and times with standards and conventions.
- Packed Data Values
  - Conventions for packing numeric data to save space have some subtleties.
- Missing Data Values
  - To indicate that data values are missing, invalid, or not written, special values are conventionally used.

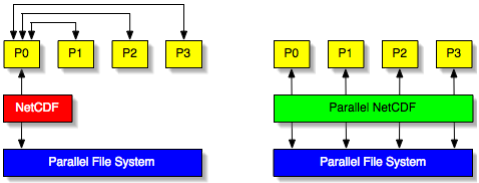
Climate and Forecast (CF) Conventions

- The Climate and Forecast (CF) conventions are metadata conventions for earth science data, intended to promote the processing and sharing of files created with the NetCDF API.
- The purpose of the CF conventions is to require conforming datasets to contain sufficient metadata that they are self-describing:
  - Each variable in the file has an associated description of what it represents, including physical units if appropriate.
  - Each value can be located in space (relative to earth-based coordinates) and time.
- The CF conventions enable users of data from different sources to decide which data are comparable and allows building applications with powerful extraction, regridding, and display capabilities.

Outline

Parallel I/O

- Parallel I/O allows each processor in a multi-processor system to read and write data from multiple processes to a common file independently.



- Data-intensive scientific applications use parallel I/O software to access files.
- In HPC, increasing demands in the I/O system can cause bottlenecks. Parallel I/O plays a fundamental role to balance the fast increase in computational power and the progress of processor architectures.
- Used properly, parallel I/O allows users to overcome I/O bottlenecks in HPC environments.

Parallel I/O in NetCDF-4

- NetCDF-4 provides parallel file access to both classic and NetCDF-4/HDF5 files.
- The parallel I/O to NetCDF-4 files is achieved through the HDF5 library while the parallel I/O to classic files is through PNetCDF.
- NetCDF-4 exposes the parallel I/O features of HDF5.
  - HDF5 provides easy-to-use parallel I/O feature.
- Parallel NetCDF uses MPI I/O to perform parallel I/O. It is a complete rewrite of the core C library using MPI I/O instead of POSIX.

Using Parallel I/O in NetCDF-4

- Special `nc_create_par` and `nc_open_par` functions are used to create/open a NetCDF file.
- The files they open are normal NetCDF-4/HDF5 files, but these functions also take MPI parameters.
- The parallel access associated with these functions is not a characteristic of the data file, but the way it was opened. The data file is the same, but using the parallel `open/create` function allows parallel I/O to take place.

```
EXTERNL int
nc_create_par(const char *path, int cmode, MPI_Comm comm,
              MPI_Info info, int *ncidp);

EXTERNL int
nc_open_par(const char *path, int mode, MPI_Comm comm,
            MPI_Info info, int *ncidp);
```

Collective and Independent Operations with Parallel I/O

- In MPI programs, I/O may be collective or independent.
  - Collective: It must be done by all processes at the same time
  - Independent: It can be done by any process at any time.
- All NetCDF metadata writing operations are collective. That is, all creation of groups, types, variables, dimensions, or attributes.
- Data reads and writes (ex. calls to `nc_put_vara_int` and `nc_get_vara_int`) may be independent (the default) or collective. To make writes to a variable collective, call the `nc_var_par_access` function.

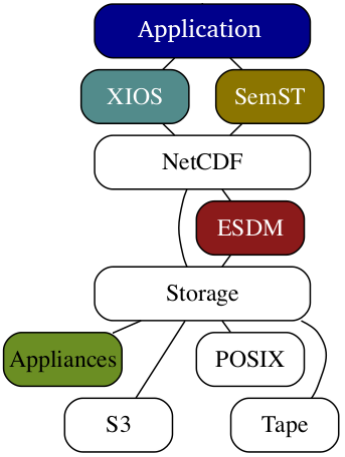
```
/* Use these with nc_var_par_access(). */
#define NC_INDEPENDENT 0
#define NC_COLLECTIVE 1

EXTERNL int
nc_var_par_access(int ncid, int varid, int par_access);
```

Outline

ESiWACE – Work Package 4

Objectives

- Support data reduction in ensembles by providing tools to carry out ensemble statistics "in-flight" and compress ensemble members.
- Hide complexity of multiple-storage tiers (middleware between NetCDF and storage) with industrial prototype backends.
- Deliver portable workflow support for manual migration of semantically important content between disk, tape, and object stores.



Earth-System Data Middleware (ESDM)

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

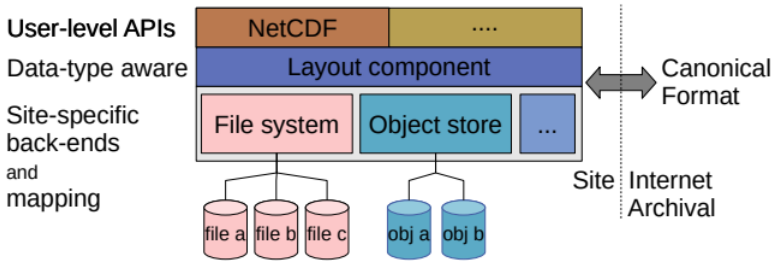
Design goals of the Earth-System Data Middleware

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

ESDM – Architecture

Key Concept: Decouple data localization decisions from science

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



ESDM – Benefits

- Expose/access the same data via different APIs
- Independent and lock-free writes from parallel applications
- No fixed storage layout
- Less performance tuning from users needed
- Exploit characteristics of different storage technology
- Multiple layouts of one data structure optimize access patterns
- Flexible namespace (similar to MP3 library)

Next Generation Interfaces (NGI)

- Towards a new I/O stack considering:
  - User metadata and workflows as first-class citizens
  - Smart hardware and software components
  - Liquid-Computing: Smart-placement of computing
  - Self-aware instead of unconscious
  - Improving over time (self-learning, hardware upgrades)
- Why do we need a new domain-independent API?
  - Other domains have similar issues
  - It is a hard problem approached by countless approaches
  - Harness RD&E effort across domains

Smart Compression

- The main purpose of compression methods is to shrink data and to save storage space.
- Compression methods also possess a huge potential to reduce the gap between computational power and I/O performance.
  - Often, after compression, less data has to be moved.
- Many modern file formats, in particular HDF5 and NetCDF-4, provide native support for compression.
  - Beneficial in climate science, where data amounts are huge and are growing constantly.
- The compression algorithms used in HDF5 and NetCDF-4 are lossless and do not meet the requirements of climate science to full extent.

Scientific Compression Library (SCIL)

- In ESiWACE, SCIL is integrated into ESDM to allow scientists to specify the data properties on datasets.
  - Developed in the AIMES Project[2].
  - The main purpose of SCIL is compression/decompression of scientific data, especially, of climate modeling data.
  - Uses different third party compression libraries as well as specifically developed lossy and lossless compression methods.
  - Separates concern of data accuracy and choice of algorithms.
  - Users specify necessary accuracy and performance parameters.
  - Metacompression library makes the choice of algorithms.
  - Supports new algorithms.

[2] Advanced Computation and I/O Methods for Earth-System Simulations (AIMES) Project.

Bibliography I

Sihem Amer-Yahia, Vincent Leroy, Alexandre Termier, Martin Kirchgessner, and Behrooz Omidvar-Tehrani.
Interactive Data-Driven Research: the place where databases and data mining research meet.
Research Report RR-LIG-049, LIG, 2015.
Les rapports de recherche du LIG - ISSN : 2105-0422.

E. Wes Bethel, Andrew C. Bauer, Brad Whitlock, Matthew Wolf, Burlen Loring, and Silvio Rizzi.
In Situ Analysis and Visualization with SENSEI, Accessed in August 2020.

Lonnie Crosby.
Parallel I/O Techniques and Performance Optimization, 2012 (Accessed in August 2020).

German Climate Computing Center (DKRZ).
ESiWACE – For Future Exascale Weather and Climate Simulations, Accessed in August 2020.

Max Hailperin and Gustavus Adolphus College.
Operating Systems and Middleware: Supporting Controlled Interaction.
Online version, 1.3.1 edition, 2019.

Stijn Heldens, Pieter Hijma, Ben Van Werkhoven, Jason Maassen, Adam S. Z. Belloum, and Rob V. Van Nieuwpoort.
The landscape of exascale research: A data-driven literature analysis.
ACM Comput. Surv., 53(2), March 2020.

Julian Kunkel, Michael Kuhn, and Thomas Ludwig.
Exascale Storage Systems – An Analytical Study of Expenses.
Supercomputing Frontiers and Innovations, pages 116–134, 06 2014.

Bibliography II


Julian Kunkel and Luciana Pedro.
Potential of I/O Aware Workflows in Climate and Weather.
Supercomputing Frontiers and Innovations, 7(2), 2020.


Julian Kunkel.
Simulation of Parallel Programs on Application and System Level.
Phd thesis, Universität Hamburg, 07 2013.


Rob Latham, Rob Ross, Brent Welch, and Katie Antypas.
Parallel I/O in Practice, Accessed in August 2020.

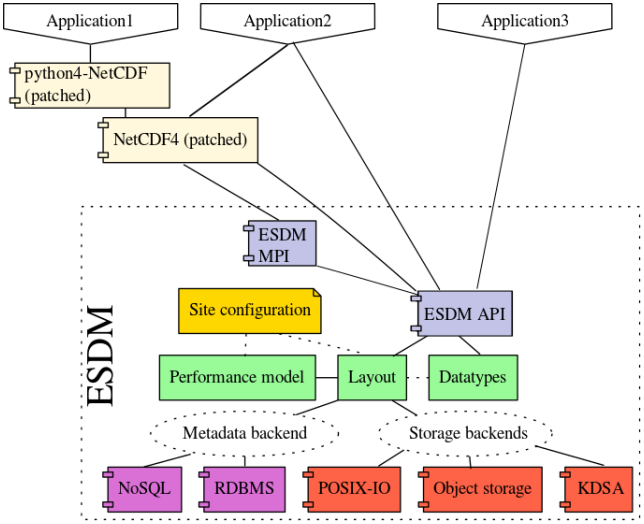

UCAR/Unidata.
Network Common Data Form (NetCDF), Accessed in August 2020.

The ESiWACE1/2 projects have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No **675191** and No **823988**

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

Appendix

ESDM Architecture



ESDM Architecture and Interactions