Computer Science Department

HPS

University of Reading

Community Development of
Next Generation Semantic Interfaces

Image under free license (CC0)

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel

HPC-IODC/WOPSSS

2018-06-28

Copyright University of Reading

Outline

Example: A Software Stack for NWP/Climate

University of Reading

- Domain semantics
  - XIOS writes independent variables to one file each
  - 2nd servers for performance reasons
- Why user side servers besides data model
  - Performant mappings to files are limited
    - Map data semantics to one "file"
    - File formats are notorious inefficient
  - Domain metadata is treated like normal data
    - Need for higher-level databases
  - Interfaces focus on variables but lack features
    - Workflows
    - Information life cycle management

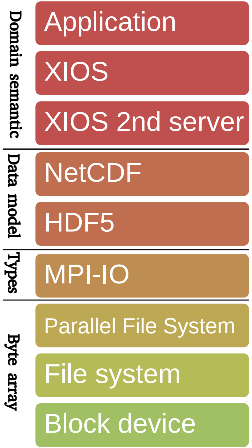


Figure: Typical I/O stack

Critical Discussion

University of Reading

Questions from the storage users' perspective

- Why do I have to organize the file format?
  - It's like taking care of the memory layout of C-structs
- Why do I have to convert data between storage paradigms?
  - Big data solutions typically do not require this step!
- Why must I provide system specific performance hints?
  - It's like telling the compiler to unroll a loop exactly 4 times
- Why is a file system not offering the consistency model I need?
  - My application knows the required level of synchronization

Being a user, I would rather code an application?

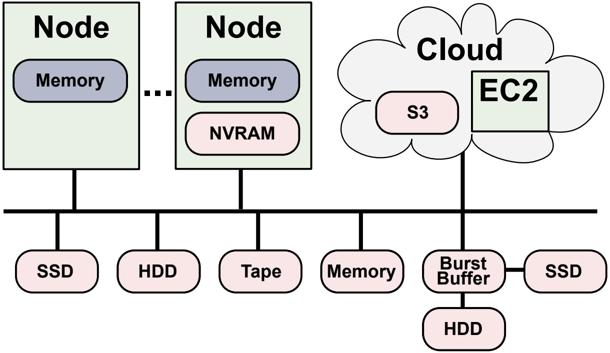
Challenges Faced by HPC I/O

University of Reading

- Difficulty to analyze behavior and understand performance
  - Unclear access patterns (users, sites)
- Coexistence of access paradigms in workflows
  - File (POSIX, ADIOS, HDF5), SQL, NoSQL
- Semantical information is lost through layers
  - Suboptimal performance, lost opportunities
  - All data treated identically (up to the user)
- Re-implementation of features across stack
  - Unpredictable interactions
  - Wasted resources
- Restricted (performance) portability
  - Optimizing each layer for each system?
  - Users lack technological knowledge for tweaking
- Utilizing the future storage landscapes
  - No performance awareness, manual tuning and mapping to storage needed

Future Systems: Coexistence of Storage Systems

University of Reading



- We shall be able to use all storage technologies concurrently
  - Without explicit migration etc. put data where it fits
  - Administrators just add a new technology (e.g., SSD pool) and users benefit

Alternative Software Stack

Some Examples

- High-level abstractions: Dataclay, Dataspaces, Mochi
- Data models: ADIOS, HDF5, NetCDF, VTK
- Standard API across file formats: Silo, VTK, CDI, (HDF5)
- Low-level libraries: SIONlib, PLFS
- Storage interfaces: MPI-IO, POSIX, vendor-specific (e.g., CLOVIS), S3
- Big-data: HDFS, Spark, Flink, MongoDB, Cassandra
- Research: Countless storage system prototypes every year

Standardization Attempts

University of Reading

Promising

- Container storage interface (community driven / involves companies)
- Cloud Data Management Interface (SNIA driven)
- pmem.io (good candidate for persistent memory programming)
- HDF5 (towards a de-facto standard interfaces)

How about HPC?

- MPI-IO (partially successful)
- Exascale10/EOFS (failed)
- *Various* earlier attempts that failed

Thoughts

University of Reading

I believe we must re-architect the IO stack **together** to

- Enable smart hardware and software components
- Optimize storage and compute together
- Deal with scientific metadata and workflows as first-class citizens
- Become self-aware instead of unconscious
- Enable self-learning and improving system behavior over time
- Allow schedulers to generate optimized plans
  - **LiquidComputing**: Running pieces on storage, compute, IoT, network

Constraints

- The process should be steered by a standard and open forum
- Open ecosystem for any vendor, research, ...

Outline

A Potential Approach in the Community: Following MPI

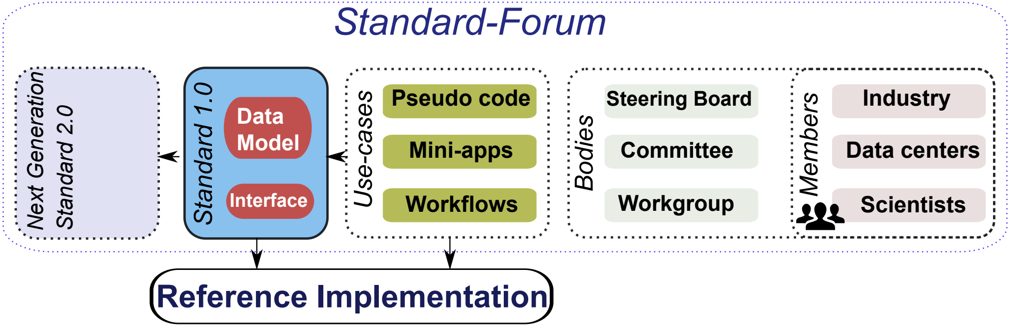
University of Reading

- The **standardization** of a high-level *data model & interface*
  - Targeting data intensive and HPC workloads
  - Lifting semantic access to a new level
  - To have a future: must be beneficial for Big Data + Desktop, too
- Development of a reference implementation of a **smart runtime system**
  - Implementing key features
- Demonstration of benefits on socially relevant data-intense apps

Development of the Data Model and API

University of Reading

- Establishing a Forum similarly to MPI
- Define data model for HPC
- Open board: encourage community collaboration



Outline

University of Reading

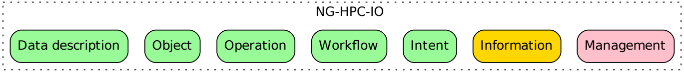
A Pragmatic View

University of Reading

- Take existing data model like VTK (or HDF5) as baseline
- With a hint of:
  - Scientific metadata handling
  - Workflow and processing interface
  - Information lifecycle management
  - Hardware model interface (hardware provides its own performance models)
- First prototype utilizes existing software stack
  - Like Cylc for workflows
  - Like MongoDB for metadata
  - Like a parallel file system (or object storage)
- Work on:
  - Scheduler for performant mapping of data/compute to storage/compute
  - A FUSE client for flexible data mappings on semantic metadata
  - Importer/Exporter tools for standard file formats
- Add magic (knowledge of experts developing APIs)
- Next prototype: move on with true implementation

Next-Generation HPC IO API Key Features

- High-level data model for HPC
  - Storage understands data structures vs. byte array
  - Relaxed consistency
- Semantic namespace and storage-aware data formats
  - Organize based on domain-specific metadata (instead of file system)
  - Support domain-specific operations and addressing schemes
- Integrated processing capabilities
  - Offload data-intensive compute to storage system
  - Managed data-driven workflows supporting events and services
  - Scheduler maps compute and I/O to hardware
- Enhanced data management features
  - Information life-cycle management (and value of data)
  - Embedded performance analysis
  - Resilience, import/export, ...



Summary

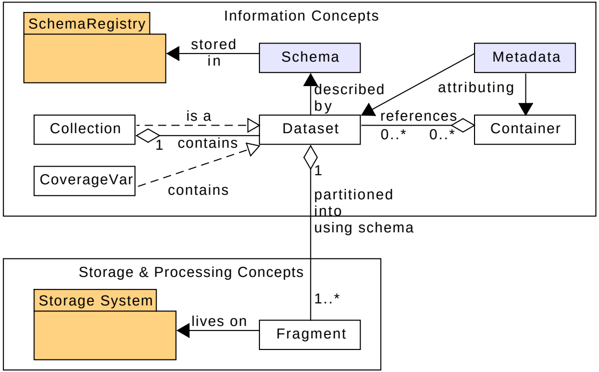
University of Reading

- The separation of concerns in the existing storage stack is suboptimal
- There is a huge potential for the next-generation interface
- Can the community work together to define next generation APIs?
- If you are interested, subscribe to: https://www.vi4io.org/mailman/listinfo/io-ngi

Appendix

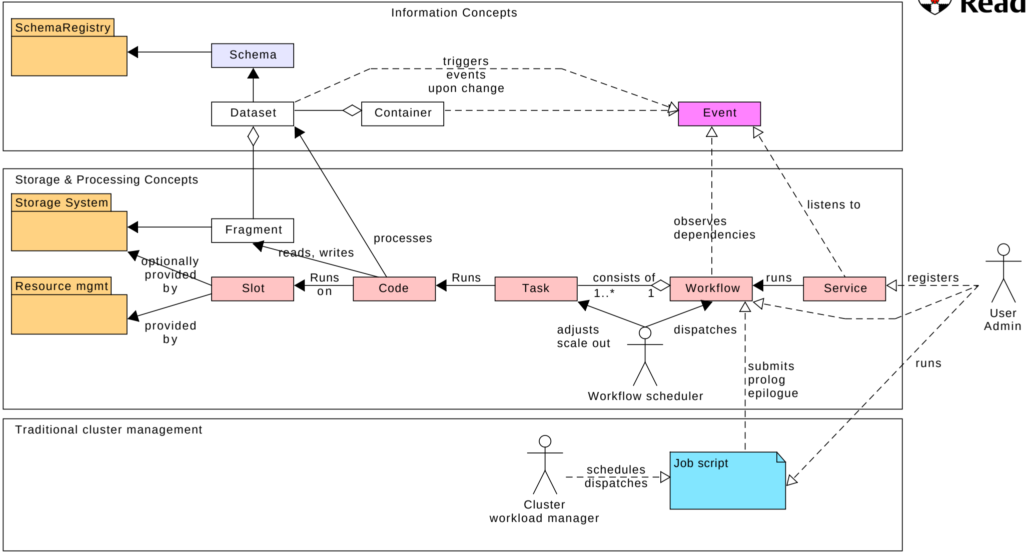
Data Model: Addressing and Metadata

University of Reading



- Data Formats are provided by schema registry
- Define addressing and high-level metadata space
- Containers, datasets, fragments

Processing Model



Outline

University of Reading

Compression Research: Involvement

University of Reading

- **Scientific Compression Library** (SCIL)
  - Separates concern of **data accuracy** and **choice of algorithms**
  - Users specify necessary accuracy and performance parameters
  - Metacompression library makes the choice of algorithms
  - Supports also new algorithms
  - Ongoing: standardization of useful compression quantities
- Development of algorithms for lossless compression
  - MAFISC: suite of preconditioners for HDF5, pack data optimally, reduces climate data by additional 10-20%, simple filters are sufficient
- Cost-benefit analysis: e.g., for long-term storage MAFISC pays of
- Analysis of compression characteristics for earth-science related data sets
  - Lossless LZMA yields best ratio but is very slow, LZ4fast outperforms BLOSC
  - Lossy: GRIB+JPEG2000 vs. MAFSISC and proprietary software
- Method for system-wide determination of ratio/performance
  - Script suite to scan data centers...

SCIL: Supported User-Space Quantities

Quantities defining the residual (error):

absolute tolerance: compressed can become true value ± absolute tolerance

relative tolerance: percentage the compressed value can deviate from true value

relative error finest tolerance: value defining the abs tol error for rel compression for values around 0

significant digits: number of significant decimal digits

significant bits: number of significant decimals in bits

field conservation: limits the sum (mean) of field's change

Quantities defining the performance behavior:

compression throughput

decompression throughput

in MiB or GiB, or relative to network or storage speed

Aim to standardize user-space quantities across compressors!
See https://www.vi4io.org/std/compression

Ongoing Activity: Earth-Science Data Middleware

Part of the ESiWACE Center of Excellence in H2020

Design Goals of the Earth System Data Middleware

1. Understand application data structures and scientific metadata
2. Flexible mapping of data to multiple storage backends
   - Placement based on site-configuration + performance model
   - Site-specific optimized data layout schemes
3. Relaxed access semantics, tailored to scientific data generation
4. A configurable namespace based on scientific metadata

EU funded Project: ESiWACE

University of Reading

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

- Representing the European community for
  - climate modelling and numerical weather simulation
- 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 highest affordable resolution
- Funding via the European Union's Horizon 2020 program (grant #675191)

http://esiwace.eu

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

Architecture