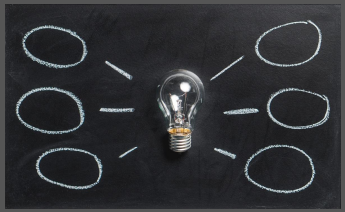
Computer Science Department

HPS

University of Reading

Towards Intelligent Storage Systems



Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel

Computer Science Workshop

2018-05-09

High-Performance Storage

University of Reading

- I/O intense science requires high I/O performance
- The file systems of the data center DKRZ offer about 700 GiB/s throughput
  - However, I/O operations are typically inefficient: Achieving 10% peak is good
  - Unfortunately, prediction of performance is barely possible
- Various factors influence I/O performance
  - Application's access pattern and usage of storage interfaces
  - Technology / Hardware
  - Concurrent activity – shared nature of I/O
  - Tunable optimizations deal with characteristics of storage media
  - Complex interactions of these factors
- The I/O hardware/software stack is very complex – even for experts

Opportunities

University of Reading

- Chances for tools and method development for:
  - Particularly: Performance models and machine learning
  - Predicting performance, identification of slow performance
  - Diagnosing causes
  - Prescribing tunables/settings
- Chance to develop more suitable interfaces!
  - Avoiding pitfalls of current interfaces
  - Harnessing smart data management

Outline

University of Reading

Peeking at the Current I/O Stack – System Perspective

University of Reading

- Coexistence of access paradigms
  - File (POSIX, ADIOS, HDF5), SQL, NoSQL
- Semantical information is lost through layers
  - Suboptimal performance
- Reimplementation of features across stack
  - Unpredictable interactions
  - Wasted ressources
- Restricted (performance) portability
  - Optimizing each layer for each system?

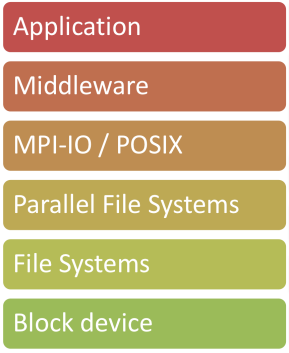


Figure: Example I/O stack

Limitations of the current software stack

University of Reading

Platform

1. Zoo of interfaces
2. Low-level storage APIs
3. Loss of semantical information
4. Interference of applications / lack of coordination
5. All data treated identically

Software

1. Explicit workflows
2. Unclear access patterns (users, sites)
3. No performance awareness, manual tuning needed
4. Users lack technological knowledge for tweaking
5. Manual tiering (or with file-grained policies)

Zoo of Interfaces

University of Reading

Multitude of data models

- POSIX File: Array of bytes
- HDF5: Container like a file system
  - Dataset: N-D array of a (derived) datatype
  - Rich metadata, different APIs (tables)
- Database: structured (+arrays)
- NoSQL: document, key-value, graph, tuple

Choosing the right interface is difficult – a workflow may involve several

Properties / qualities

- Namespace: Hierarchical, flat, relational
- Access: Imperative, declarative, implicit (`mmap()`)
- Concurrency: Blocking vs. non-blocking
- Consistency semantics: Visibility and durability of modifications

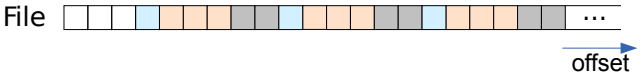
Low Level Data Access

University of Reading

Applications work with (semi)structured data

- Vectors, matrices, n-Dimensional data

A file is just a sequence of bytes!



Applications/Programmers must serialize data into a flat namespace

- Uneasy handling of complex data types
- Mapping is performance-critical (on HDDs)
- Vertical data access unpractical (e.g., to to pick a slice of multiple files)

Loss of Semantical Information

University of Reading

Information hidden from file systems

- Data types
- Data semantics
- Value of data
- Type: Checkpoint, computed, original, logfile
- Data lifecycle: production, usage, deletion

Characteristics can even vary within a file, e.g. for metadata

Storage systems could use this information for

- Improving performance: Automatic tiering, caching, replication
- Simplifying management: ILM, offering alternative data views
- Correctness: Ensuring data consistency

Manual Tuning

University of Reading

- There are many options to tweak the I/O-stack
  - API: `posix_fadvise()`, HDF5 properties, open flags, cache size
  - Via command line: `lfs setstripe`
  - Setup/initialization of a storage system
  - Mounting options and procfs settings
- Many options are of technical nature
  - Performance gain/loss depend on hardware, software
  - Specific to file system, API (MPI, POSIX, HDF5)
  - Many types of hints/tweaks are not portable
- Performance loss forces us to use these optimization

Critical Discussion

University of Reading

Questions from the 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?
- 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
- Why can't I rely on a correct implementation of the consistency model?
  - Parallel file systems have performance issues with most models

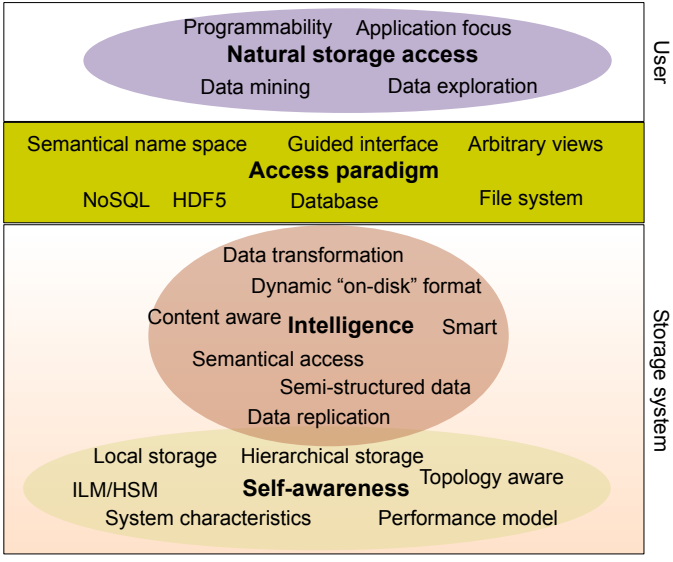
Being a user, I would rather code an application?

Outline

University of Reading

Personal Vision: Intelligent Storage Systems

University of Reading



Semantic Interfaces

University of Reading

Guiding vs. automatism vs. technical hints

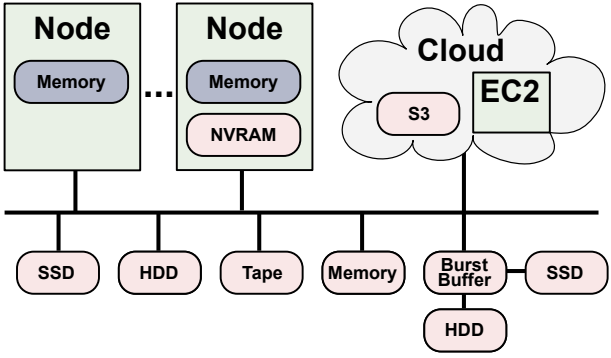
- Users provide semantic information to guide an intelligent system.
- The I/O stack may exploit this information or not.
- Systems could improve (via ML) over time by using the information better.

Semantic information which could be provided by users

- Data types
- Metadata like meaning
- Relations between data
- Lifecycle (especially usage)

Coexistence of Storage Systems vs. Tiering

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
  - Should be steered by a standard and open interface
  - Open ecosystem for any vendor...

Additional Responsibilities of Storage System

University of Reading

- Mapping of data structures
- Flexible semantics
- Compute offloading, see success of big data tools
- Tight integration of workflows
- Advanced performance assessment
- Namespace based on metadata
- Management tools
- ...

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

University of Reading

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

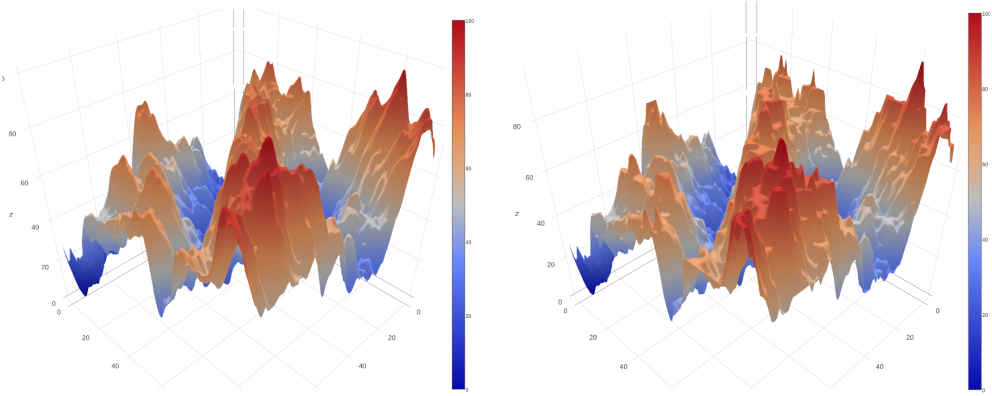
in MiB or GiB, or relative to network or storage speed

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

SCIL Provides Typical Synthetic Data

University of Reading

Example: Simplex (options 206, 2D: 100x100 points)



Right picture compressed with Sigbits 3bits (ratio 11.3:1)

Ongoing Activity: Earth-Science Data Middleware

University of Reading

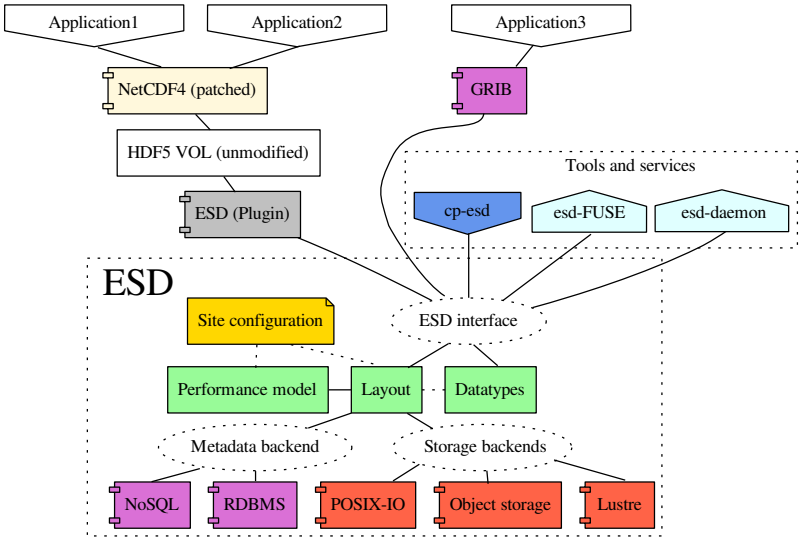
Part of the ESiWACE Center of Excellence in H2020

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
3. Placement based on site-configuration + performance model
4. Site-specific optimized data layout schemes
5. Relaxed access semantics, tailored to scientific data generation
6. A configurable namespace based on scientific metadata

Architecture

University of Reading



Outline

University of Reading

A Potential Approach in the Community: Following MPI

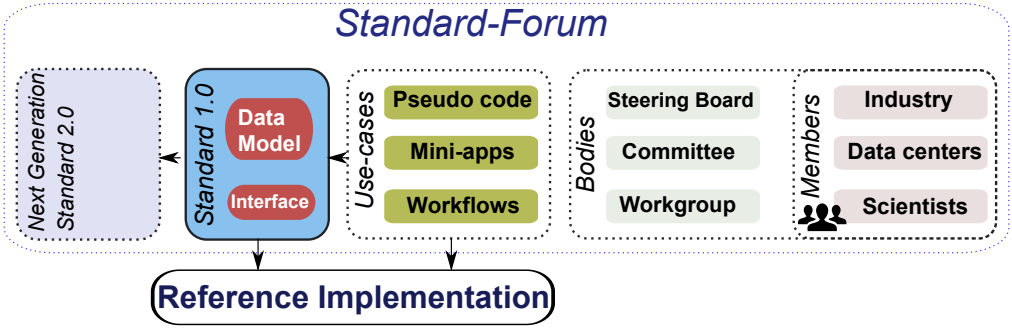
University of Reading

- The **standardization** of a high-level *data model & interface*
  - Targeting data intensive and HPC workloads
  - Lifting semantic access to a new level
- 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
  - To have a future: must be beneficial for Big Data + Desktop, too
- Open board: encourage community collaboration



API Key Features

University of Reading

- High-level data model for HPC
  - Storage understands data structures vs. byte array
  - Relaxed consistency
- Semantic namespace
  - Organize based on domain-specific metadata (instead of file system)
  - Support domain-specific opperations and addressing schemes
- Integrated processing capabilities
  - Offload data-intensive compute to storage system
  - In-situ/In-transit workflows
- Workflow management
  - Managed data-driven workflow
- Performance-portability
  - Guided interfaces: Intents vs. technical hints
- Enhanced data management features
  - Embedded performance analysis
  - Resilience, import/export, ...

Summary

University of Reading

- Parallel I/O is complex
  - System complexity and heterogeneity increases significantly
  - HPC users (scientists) and data centers need methods and tools
- Abstract semantic interfaces combined with ML will be a game changer
- Intelligent systems will increase insight and reduce the burden for users
- I will attempt a community strategy
- High performance storage web page: https://hps.vi4io.org

Appendix

High Performance Storage

University of Reading

- Efficient I/O
  - Performance analysis methods, tools and benchmarks
  - Modeling of performance and costs
  - Optimizing parallel file systems and middleware
  - Tuning of I/O: Prescribing settings
  - Management of workflows
  - Intelligent storage
- Interfaces: towards domain-specific solutions
- Data reduction: compression library, algorithms, methods, re-computation

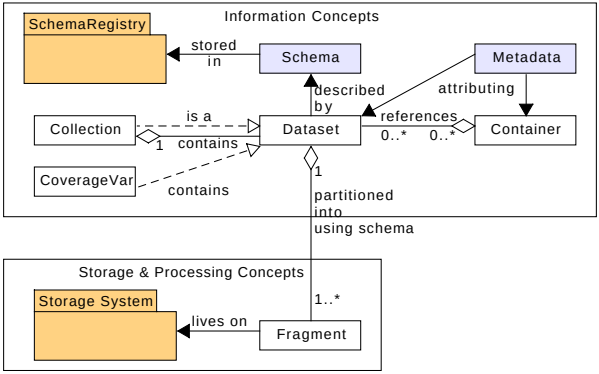
Other Research Interests

University of Reading

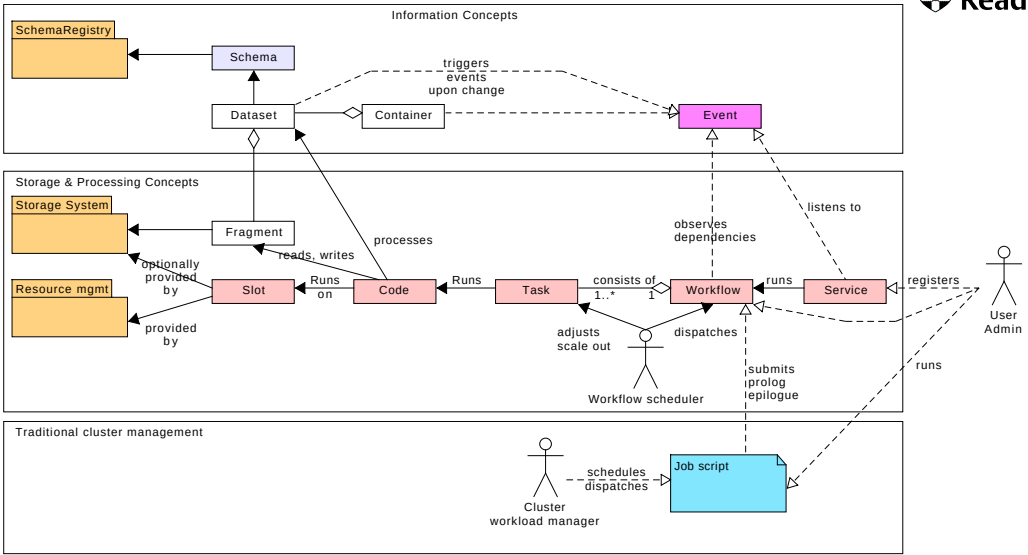
Scientific and High Performance Computing

- Big Data Analytics
- Domain-specific languages
- Cost-efficiency and data centers
- Scientific programming
- Efficient teaching

Data Model


Information Concepts
SchemaRegistry
stored in
Schema
Metadata
described by
attributing
Collection
is a
Dataset
references
Container
1
contains
0..*
0..*
CoverageVar
contains
1
partitioned into
using schema
Storage & Processing Concepts
Storage System
1..*
lives on
Fragment

Processing Model



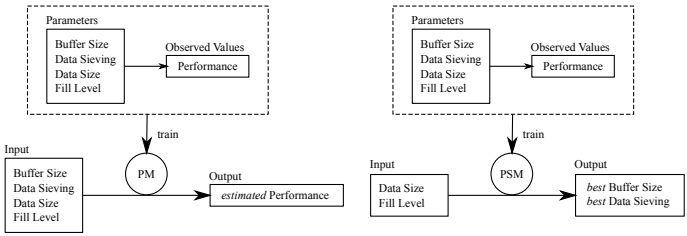
Predicting Non-Contiguous I/O Performance

University of Reading

Goal: Predict storage performance based on several parameters and tunables

Alternative models

- Predict performance based on parameters
- Predict best (data sieving) settings



(a) Performance Model (b) Parameter Setting Model

Figure: PM provides a perf. estimate, whereas PSM provides the tunable variable parameters to

Extracting Knowledge

University of Reading

- Rules can be easily extracted from decision trees
- Consider a performance prediction in three classes
- Rules (this is common sense for I/O experts)
  - Small fill levels and data sizes are slow
  - Large fill levels achieve good performance
- Surprising anomaly: smaller fill level, large access sizes are slower than medium

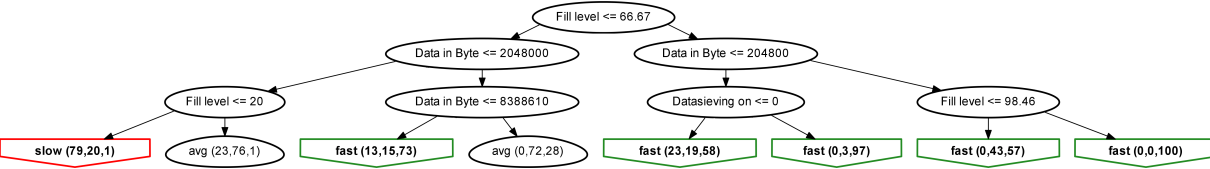


Figure: First three levels of the CART classifier rules for three classes slow, avg, fast ($[0, 25]$, $(25, 75]$, > 75 MB/s). The dominant label is assigned to the leaf nodes – the probability for each class is provided in brackets.

Prescriptive Analysis: Learning Best-Practises for DKRZ

University of Reading

- Performance benefit of I/O optimizations is non-trival to predict
- Non-contiguous I/O supports data-sieving optimization
  - Transforms non-sequential I/O to large contiguous I/O
  - Tunable with MPI hints: enabled/disabled, buffer size
  - Benefit depends on system AND application
- Data sieving is difficult to parameterize
  - What should be recommended from a data center's perspective?

System-Wide Defaults

University of Reading

- Comparing a default choice with the best choice
- All default choices achieve 50-70% arithmethic mean performance
- Picking the best default default for stripe count/size: 2 servers, 128 KiB
  - 70% arithmetic mean performance
  - 16% harmonic mean performance ⇒ some choices result in slow performance

Default Choice			Best	Worst	Arithmethic Mean			Harmonic Mean	
Servers	Stripe	Sieving	Freq.	Freq.	Rel.	Abs.	Loss	Rel.	Abs.
1	128 K	Off	20	35	58.4%	200.1	102.1	9.0%	0.09
1	2 MiB	Off	45	39	60.7%	261.5	103.7	9.0%	0.09
2	**128 K**	**Off**	87	76	**69.8%**	209.5	92.7	8.8%	0.09
2	2 MiB	Off	81	14	72.1%	284.2	81.1	8.9%	0.09
1	128 K	On	79	37	64.1%	245.6	56.7	15.2%	0.16
1	2 MiB	On	11	75	59.4%	259.2	106.1	14.4%	0.15
2	**128 K**	**On**	80	58	**68.7%**	239.6	62.6	**16.2%**	0.17
2	2 MiB	On	5	74	62.9%	258.0	107.3	14.9%	0.16

Table: Performance achieved with any default choice

Applying Machine Learning

University of Reading

- Building a tree with different depths
- Even small trees are much better than any default
- A tree of depth 4 is nearly optimal; avoids slow cases

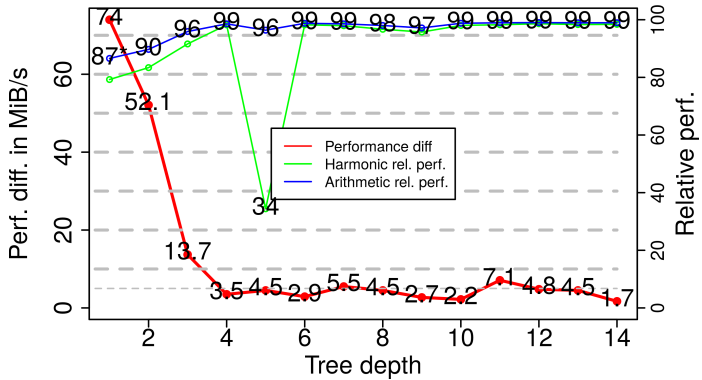


Figure: HPS perf. difference between learned and best choices, by maximum tree depth, for DKRZ's

Decision Tree & Rules

University of Reading

Extraction of knowledge from a tree

- For writes: Always use two servers; For holes below 128 KiB ⇒ turn DS on, else off
- For reads: Holes below 200 KiB ⇒ turn DS on
- Typically only one parameter changes between most frequent best choices