Research Achievements and Perspectives for HPC and Earth Science

Julian M. Kunkel

2017-06-14

Outline

High-Performance Computing (HPC)

Definitions

- HPC: Field providing massive compute resources for a computational task
  - Task needs too much memory or time on a normal computer
  - $\Rightarrow$ Enabler of complex scientific simulations, e.g., weather, astronomy
- Supercomputer: aggregates power of 10,000 compute devices
- File system: provides a hierarchical namespace and “file” interface
- Parallel I/O: multiple processes can access distributed data concurrently

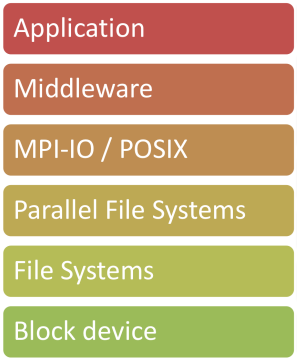
Example Supercomputer of DKRZ: Mistral

- Compute: 3,000 dual socket nodes
- Storage: 52 Petabyte (ca. 10,000 HDDs, 300 I/O servers)
- Cost for I/O system: 6 Million Euro

The I/O Stack

- Parallel application
  - Is distributed across many nodes
  - Has a specific access pattern for I/O
  - May use several interfaces
    File (POSIX, ADIOS, HDF5), SQL, NoSQL
- Middleware provides high-level access
- POSIX: ultimately file system access
- Parallel file system: Lustre, GPFS, PVFS2
- File system: EXT4, XFS, NTFS
- Operating system: (orthogonal aspect)

The layers provide optimization strategies and tunables



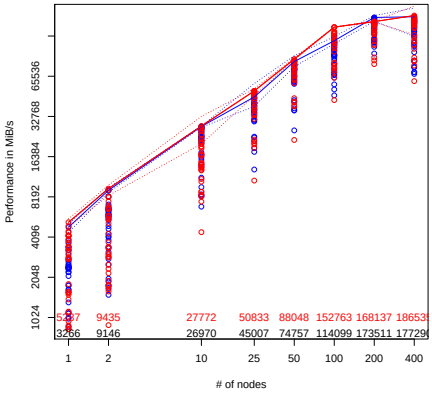
Example I/O stack

Parallel I/O

- I/O intense science requires good I/O performance
- DKRZ file systems offer about 700 GiB/s throughput
  - However, I/O operations are typically inefficient: Achieving 10% of peak is good
  - Unfortunately, prediction of performance is barely possible
- Influences on I/O performance
  - Application's access pattern and usage of storage interfaces
  - Communication and slow storage media
  - 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
- Chances for tools and method development:
  - Diagnosing causes
  - Predicting performance, identification of slow performance
  - Prescribing tunables/settings

Illustration of Performance Variability

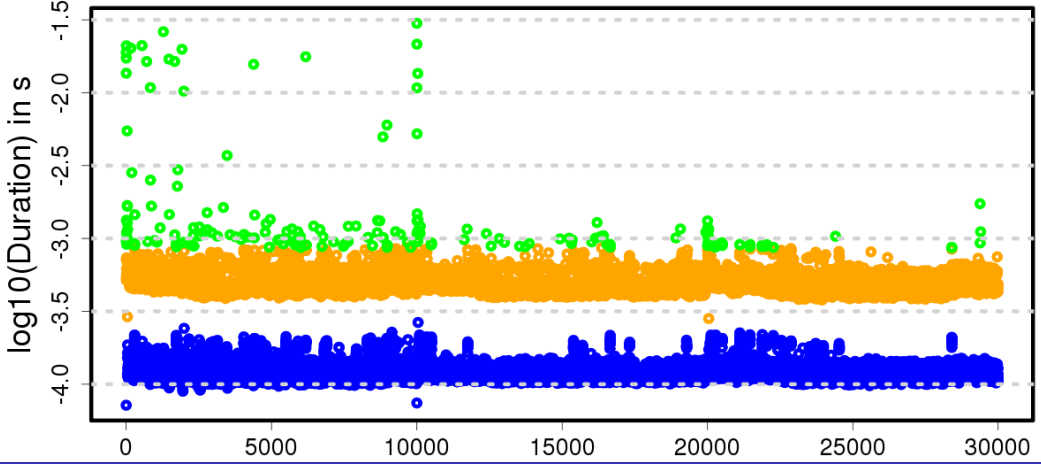
- Best-case benchmark: optimal application I/O
  - Independent I/O with 10 MiB chunks of data
  - Real-world I/O is sparse and behaves worse
- Configurations vary:
  - Number of nodes the benchmark is run
  - Processes per node
  - Tunable: stripe size, stripe count
  - Read/Write accesses
- Optimal performance:
  - Small configuration: 6 GiB/s per node
  - Large configurations: 1.25 GiB/s per node
- Best setting depends on configuration!



A point represents one configuration

Illustration of Performance Variability (2)

- Rerunning the same operation (access size, ...) leads to performance variation
- Individual measurements – 256 KiB sequential write (outliers purged)



Outline

Research Activities & Interest

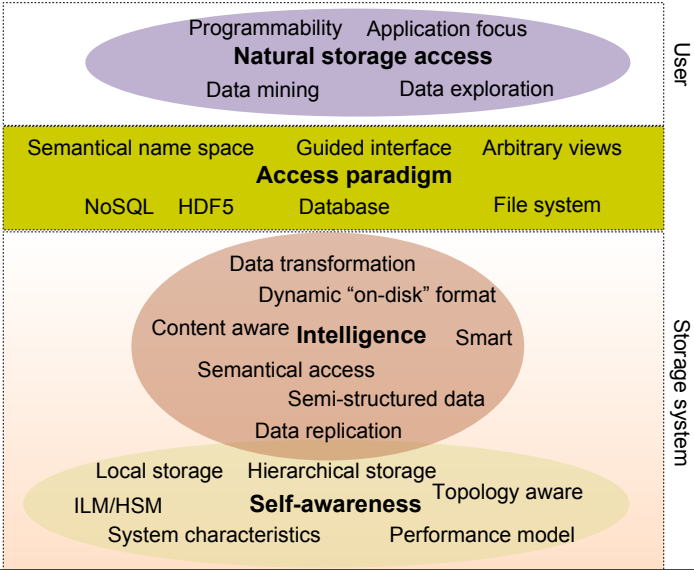
High-performance storage for HPC

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

Other research interests

- Cost-efficiency for data centers in general
- Domain-specific DSL for Icosahedral climate models
- Application of statistics and machine learning (e.g., for humanities)

Personal Vision: Towards Intelligent Storage Systems and Interfaces



Support Activities

- Community building
  - Exascale10
  - European Open File System (EOFS) `https://www.eofs.eu/`
  - The Virtual Institute for I/O `https://www.vi4io.org`
- Awareness: towards the IO-500 list
- Teaching:
  - Online teaching platform (ICP project)
  - Towards a HPC certification program (PeCoH project)
- Standardization: e.g., compression interfaces (AIMES project)

Outline

Performance Analysis

Problem

Assessing observed time for I/O is difficult

What best-case performance can we expect?

Support for analysis – my involvement

- Models and simulation
  - Trivial models: using throughput + latency
  - PIOSimHD: MPI application + storage system simulator
- Tools to capture 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-real-io)
- **Statistic model** to determine likely cause based on time

I/O Modeling and Diagnosing Causes with Statistics

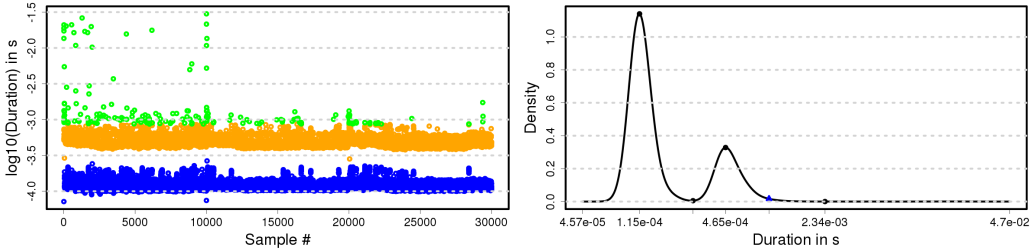
Issue

- Measuring the same operation repeatedly results in different runtime
- Reasons:
  - Sometimes a certain optimization is triggered, shortening the I/O path
  - Example strategies: read-ahead, write-behind
- Consequence: Non-linear access performance, time also depends on access size
- It is difficult to assess performance of even repeated measurements!

Goal

- Predict likely reason/cause-of-effect by just analyzing runtime
- Estimate best-case time, if optimizations would work as intended

Comparing Density Plot with the Individual Data Points

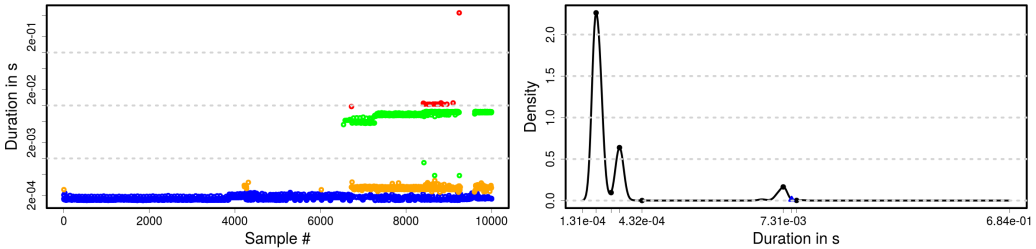


Duration for sequential reads with 256 KiB accesses (off0 mem layout)

Algorithm for determining classes (color schemes)

- Create density plot with Gaussian kernel density estimator
- Find minima and maxima in the plot
- Assign one class for all points between minima and maxima
- Rightmost hill is followed by cutoff (blue) close to zero $\Rightarrow$ outliers (unexpected slow)

Write Operations



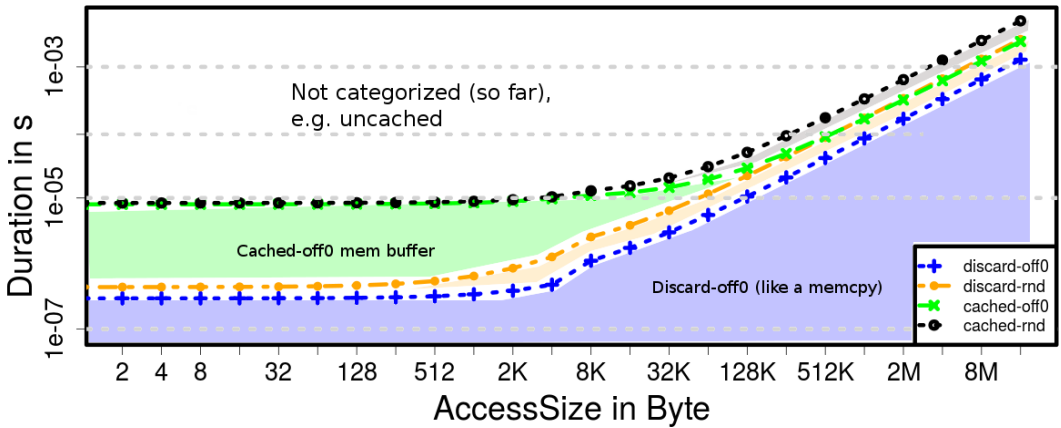
Results for one write run with sequential 256 KiB accesses (off0 mem layout).

Known optimizations for write

- Write-behind: cache data first in memory, then write back
- Write back is expected to be much slower

This behavior can be seen in the figure !

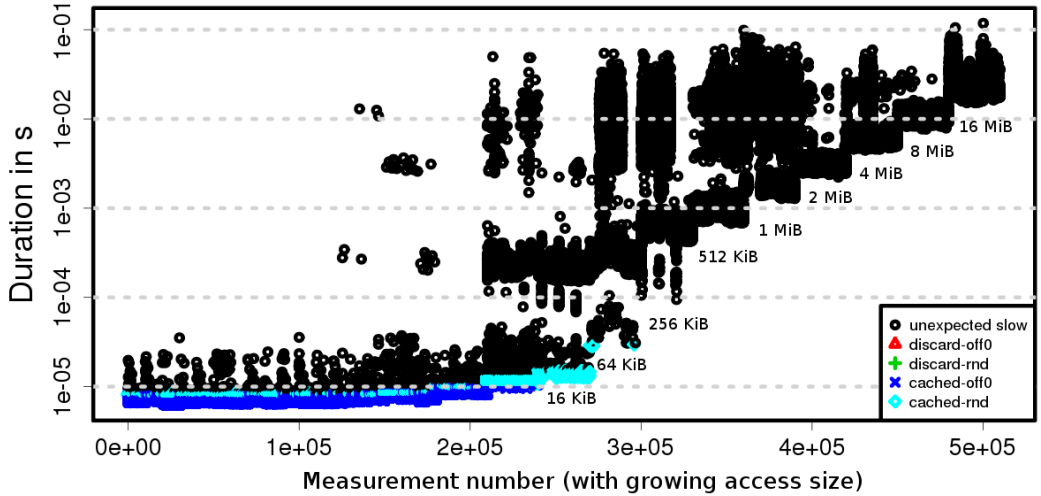
Resulting Performance Models for Read



Read models predicting caching and memory location.

Using the Model to Identify Anomalies

Using the model, the figure for reverse access shows slow down (by read-ahead)



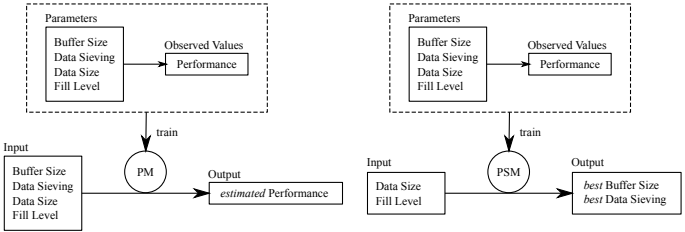
Outline

Predicting Non-Contiguous I/O Performance

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

PM provides a perf. estimate, whereas PSM provides the “tunable” variable parameters to achieve it

Validation on Data of the WR Cluster

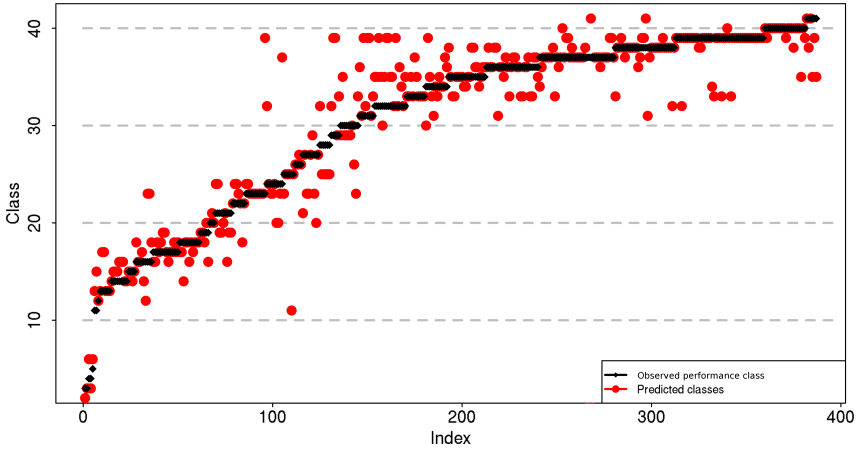
- Apply k-fold cross-validation
  - Split data into training set and validation set
  - Train model with all (k-1) folds and evaluate it on 1 fold
  - Repeat the process until all folds have been predicted
- A baseline model is the arithmethic mean performance (54.7 MiB/s)
  - Achieves an arithmethic mean error of 28.5 MiB/s
- Linear models yield a mean error of $\geq$ 12.7 MiB/s

CART results

k	Performance errors in MB/s			Class errors		
	min	mean	max	min	mean	max
2	6.74	6.80	6.87	1.46	1.59	1.72
4	5.19	6.25	6.92	0.94	1.34	1.72
8	4.67	5.66	6.77	0.87	1.19	1.62

Prediction errors for training sets under *k*-fold cross-validation. Min & max refer to the folds' mean error. Values for k=3..7 lie in between

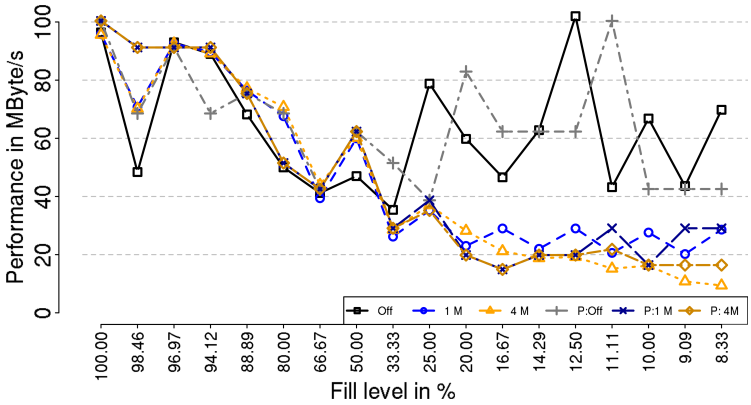
Comparing Prediction with Observation



Performance classes and error for k=2, sorted by the observed performance class. Trained by 387 instances, validated on the other 387 instances.

Comparing Prediction with Observation

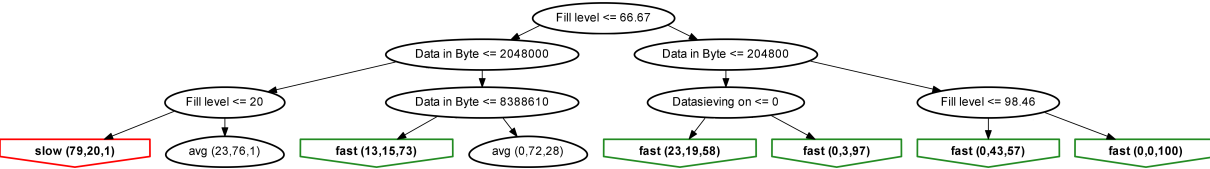
- Mispredictions due to sparse training data
- Non-linear performance behavior causes errors



Performance prediction for $d_{\text{data}} = 256\,\text{KiB}$, 387 instances

Extracting Knowledge

- 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



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

- 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?

Measured Data

- Simple single threaded benchmark, vary access granularity and hole size
- Captured on DKRZ porting system for Mistral
- Vary Lustre stripe settings
  - 128 KiB or 2 MiB
  - 1 stripe or 2 stripes
- Vary data sieving
  - Off or On (4 MiB)
- Vary block and hole size (similar to before)
- 408 different configurations (up to 10 repeats each)
  - Mean arithmetic performance is 245 MiB/s
  - Mean can serve as baseline “model”

System-Wide Defaults

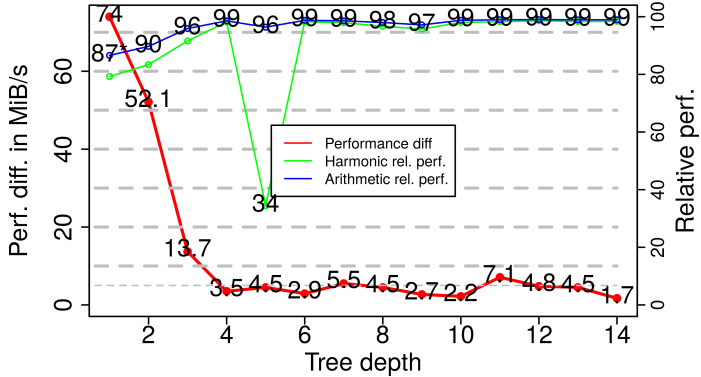
- Comparing a default choice with the best choice
- All default choices achieve 50-70% arithmethic mean performance
- Picking the best default default for stripe count/size: 2 servers, 128 KiB
  - 70% arithmetic mean performance
  - 16% harmonic mean performance ⇒ some choices result in slow performance

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

Performance achieved with any default choice

Applying Machine Learning

- 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

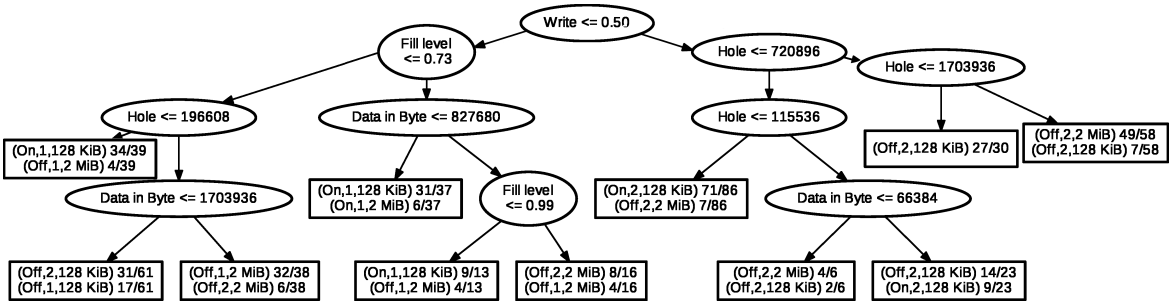


Perf. difference between learned and best choices, by maximum tree depth, for DKRZ's porting system

Decision Tree & Rules

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



Decision tree with height 4. In the leaf nodes, the settings (Data sieving, server number, stripe size) and number of instances for the two most frequent best choices

Outline

Compression Research: Involvement

- Development of algorithms for lossless compression
  - MAFISC: suite of preconditioners for HDF5, aims to pack data optimally
    Reduced climate/weather 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
- Development of the **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 a **method for system-wide determination** of ratio/performance
  - Method has been integrated into a 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 absolute tolerable error for relative 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

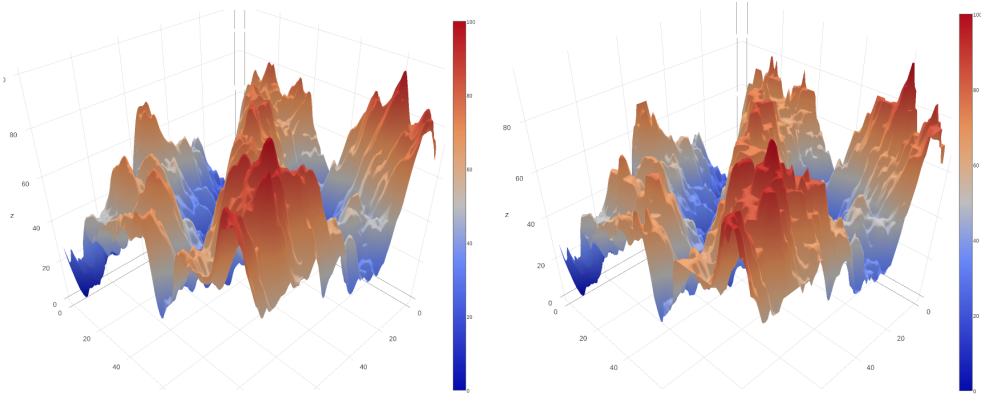
decompression throughput

- 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

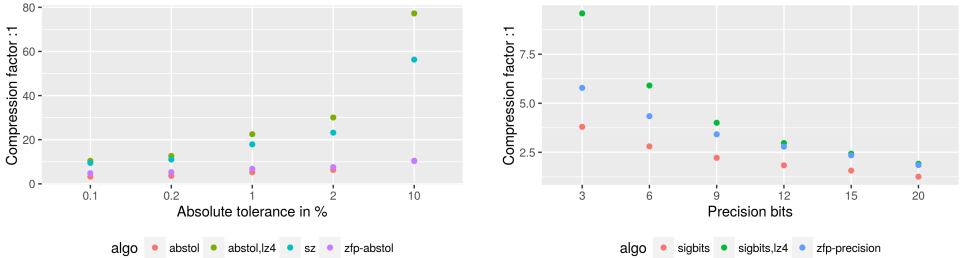
Example: Simplex (options 206, 2D: 100x100 points)



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

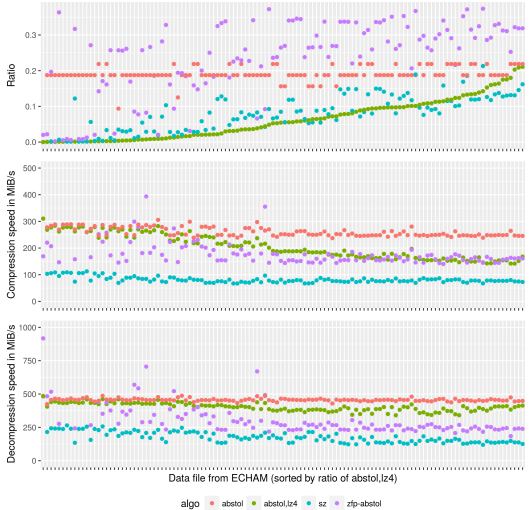
Tolerance-Based Results

- Mean compression factor across all scientific files (ECHAM5 variables)
- Factor 50:1 means space is reduced to 2% of the original size
- Note that ZFP does not always reach the set precision
  - Often the absolute and precision bit tolerance cannot be met



Results for Absolute Tolerance

Comparing algorithms using an absolute tolerance of 1% of the maximum value

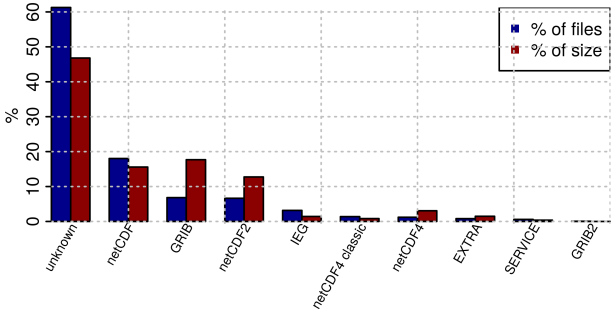


Determining Data Characteristics

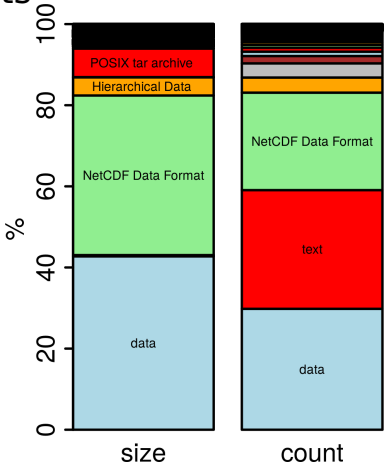
- Data characteristics:
  - Proportion of a given (scientific) file format
  - Performance behavior when accessing file data
  - Compression characteristics (ratio, speeds)
- Understanding these characteristics is useful
  - Proportions of a file format to identify relevant formats
    - Starting point for optimization of format
  - Conducting what-if analysis
    - Estimate the influence storage compression has
    - Performance expectations when applying a new strategy
- Existing studies use a manual selection of “data” for representing stored data
- Conducting analysis on representative data is non-trivial
  - What data makes up a representative data set?
  - How can we infer knowledge for all data based on the subset?
    - Based on file number/count (i.e., a typical file is like X)
    - Based on file size (i.e., 10% of storage capacity is like Y)

Example: Determine Scientific File Formats

- Notice the difference by file count and capacity



(a) Scientific file types



(b) File types according to file magic

Contribution

Goal

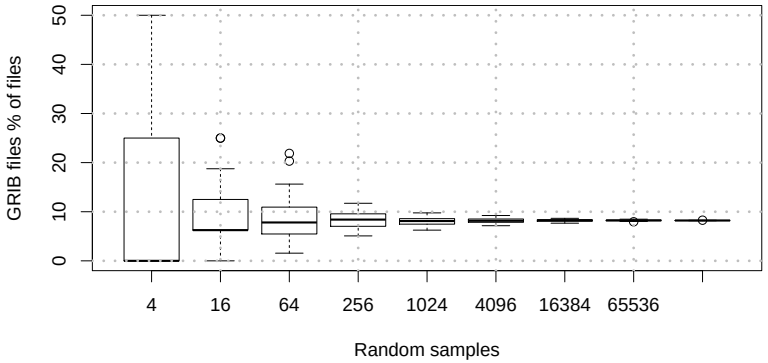
- Design a method based of statistical sampling to estimate file properties
- Conduct a simple study to investigate compression and file types

Approach

1. Scanning a large fraction of data on DKRZ file systems
   - Analyzing file types, compression ratio and speed
2. Investigating characteristics of the data set Filetype, compression ratio, ...
3. Statistical simulation of sampling approaches
   - We assume the population (full data set) is the scanned subset
4. Discuss the estimation error for several approaches

Investigating Robustness: Computing by File Count

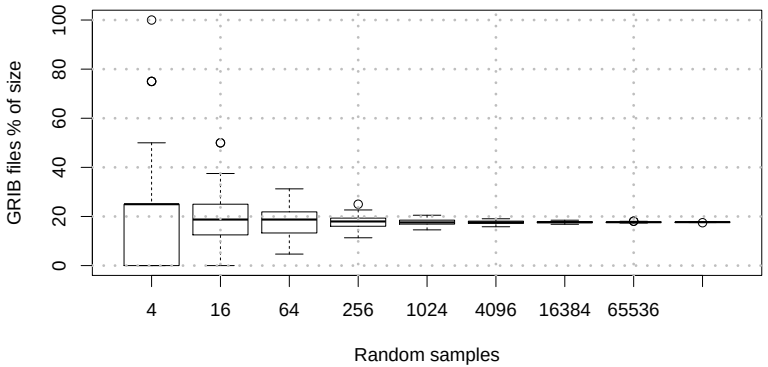
- Running the simulation 100 times to understand the variance of the estimate
- Clear convergence: thanks to Cochran's formula, the total file count is irrelevant



Simulation of sampling by file count to compute compr.% by file count

Investigating Robustness: Computing by File Size

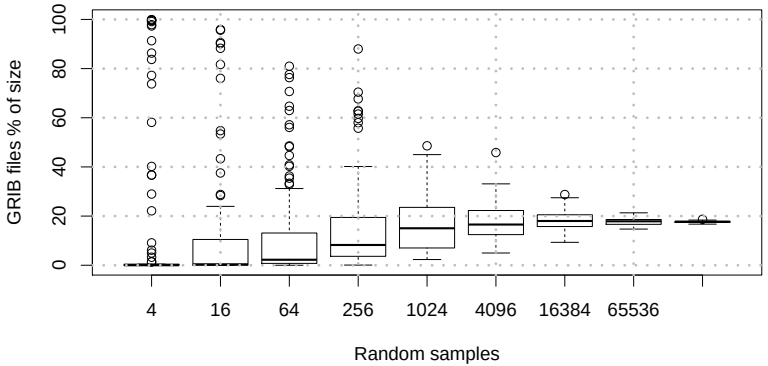
- Using the correct sampling by weighting probability with file size



Simulation of sampling to compute proportions of types by size

Investigating Robustness: Computing by File Size

- Using the WRONG sampling by just picking a simple random sample
- Almost no convergence behavior; you may pick a file with 99% file size at the end



Simulation of sampling to compute proportions of types by size

Selected Algorithms with Good Properties (out of 160+)

Algorithm	Ratio	Compr MiB/s	Decom. MiB/s
csc33-5	0.485	3.4	16.7
lzlib17-9	0.491	1.4	17.0
xz522-9	0.493	2.1	20.8
lzma938-5	0.493	2.2	24.2
brotli052-11	0.510	0.2	110.6
lzma938-2	0.526	7.9	23.1
zstd100-22	0.526	2.2	294.3
xpack2016-06-02-9	0.548	12.3	282.9
brotli052-5	0.549	16.5	156.6
xpack2016-06-02-6	0.549	16.9	278.9
zstd100-11	0.549	13.8	394.0
zstd100-2	0.574	177.6	455.3
lz4hcr131-16	0.640	3.1	1522.2
lzsse22016-05-14-16	0.640	7.7	1341.6
lz4hcr131-12	0.640	9.4	1519.5
lz4hcr131-9	0.640	17.2	1511.5
lz4hcr131-4	0.649	30.0	1477.8
lz515	0.673	229.2	858.6
density0125beta-2	0.683	419.4	496.5
pithy2011-12-24-9	0.694	305.9	1131.4
lzo1x209-1	0.726	606.7	833.7
lz4r131	0.726	469.8	1893.1
lz4fastr131-3	0.741	646.1	2001.1
lz4fastr131-17	0.772	1132.	2263.1
blosclz2015-11-10-3	0.872	494.4	2612.6
blosclz2015-11-10-1	0.900	819.4	2496.9
memcpy	1.000	4449.	4602.0

WR data

Algorithm	Ratio	Compr MiB/s	Decom. MiB/s
lzlib17-9	0.426	1.5	22.0
xz522-9	0.427	2.2	24.3
lzma938-5	0.431	2.9	29.1
lzham10-d26-1	0.445	1.4	113.3
csc33-3	0.445	6.5	23.3
brotli052-11	0.451	0.3	124.5
lzma938-0	0.473	13.0	28.2
zstd080-22	0.476	1.1	260.7
brotli052-5	0.489	18.4	165.6
zstd080-18	0.496	3.9	434.4
xpack2016-06-02-9	0.498	19.3	386.8
xpack2016-06-02-1	0.504	53.5	362.0
zstd080-5	0.511	69.4	560.8
brotli052-2	0.512	126.6	168.7
zstd080-2	0.518	220.9	594.0
zstd080-1	0.523	355.0	633.9
lzo1c209-999	0.566	13.5	939.5
lz5hc15-4	0.574	126.3	1410.1
lz515	0.576	326.9	1934.9
lz4hcr131-16	0.577	3.1	2720.6
lz4hcr131-12	0.577	12.4	2700.8
lz4hcr131-9	0.577	28.4	2670.3
lzo1b209-6	0.578	143.3	992.5
lz4r131	0.599	951.4	3037.4
lz4fastr131-3	0.603	1272.	3215.6
pithy2011-12-24-3	0.613	1787.	3535.2
lz4fastr131-17	0.614	1904.	3610.3

DKRZ data

Outline

Perspectives at Reading

Continuation of ongoing research tracks

- Parallel I/O $\Rightarrow$ efficient I/O
  - Understanding behavior, costs and options
  - Co-design of future I/O interface
  - Data reduction techniques
  - Performance portability
- Big data in earth science and humanities
- Domain specific languages

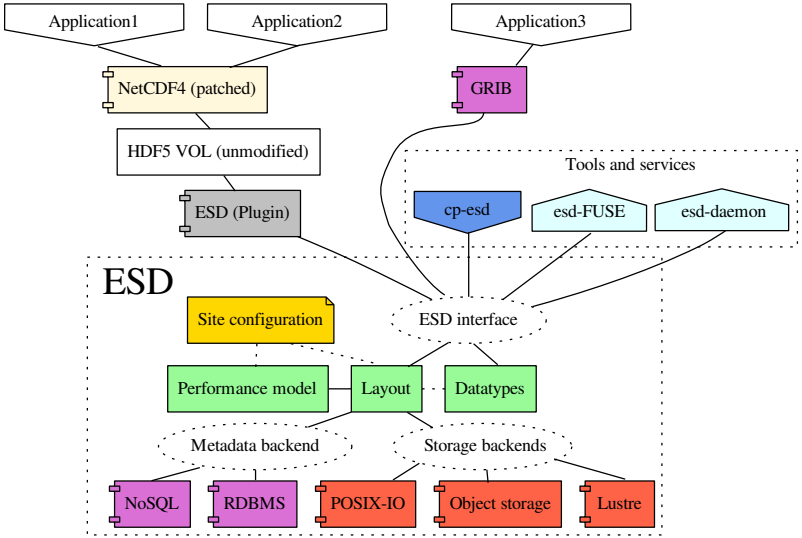
Ongoing Activity: Earth-Science Data Middleware

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
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



Summary

- Parallel I/O is complex
  - System complexity and heterogeneity increases significantly
  - $\Rightarrow$ Expected and measured performance is difficult to assess
  - HPC users (scientists) and data centers need methods and tools
- Tools, statistics and machine learning help with key aspects:
  - Diagnosing causes and identify anomalies
  - Predicting performance
  - Prescribing best practices
- I work towards intelligent systems to increase insight and ease the burden for users

Appendix