Department of Computer Science

HPS

University of Reading

Smart Processing for Extreme Data

Limitless Storage
Limitless Possibilities

https://hps.vi4io.org

Julian M. Kunkel

Georg-August-Universität Göttingen

2020-12-18

Outline

University of Reading

Research Activities & Interest

High-performance storage for HPC

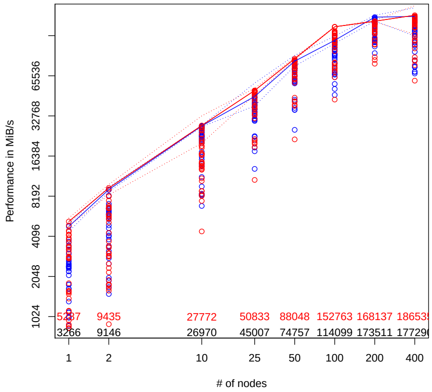
- Efficient I/O
  - Performance analysis methods, tools and benchmarks
  - Optimizing parallel file systems and middleware
  - Modeling of performance and costs
  - Tuning: Prescribing settings
  - Management of (data-driven/big data) workflows
- Data reduction: compression library, algorithms, methods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

Other research interests

- Application of big data analytics (e.g., for humanities, medicine)
- Cost-efficiency for data centers in general
- Scientific Software Engineering
- Domain-specific languages

Illustration of Performance Variability

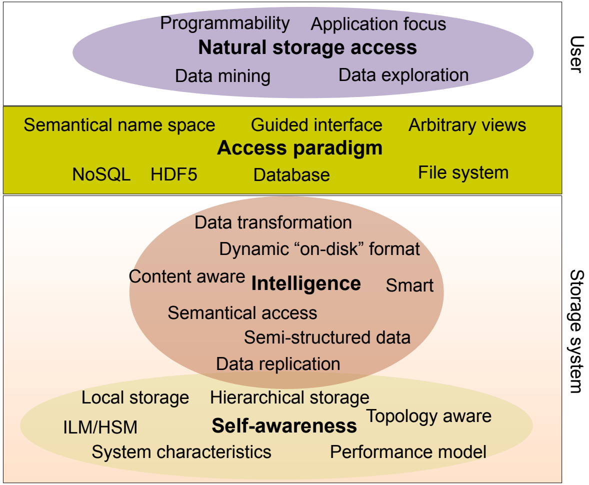
- Measured at DKRZ (max. 700 GiB/s)
- Optimal performance:
  - Small configuration: 6 GiB/s per node
  - Large configurations: 1.25 GiB/s per node
- Best-case benchmark: optimal application I/O
  - Independent I/O with 10 MiB chunks of data
  - Real-world I/O is sparse and worse
- Configurations on user-side vary:
  - Number of nodes the benchmark is run
  - Processes per node
  - Read/Write accesses
  - Tunable: stripe size, stripe count
- Best setting depends on configuration!



A point represents one configuration

Personal Vision: Towards Intelligent Storage Systems and Interfaces

University of Reading



- Abstract data interfaces
- Enhanced data management
- Integrated compute/storage
- Flexible views on data
- Smart hardware/storage
  - Self-aware systems
  - AI optimized placement
  - Bring-your-own-behavior model
- Across sites and cloud

Outline

University of Reading

Ongoing Activity

University of Reading

Earth-System Data Middleware

- Part of the ESiWACE 1+2 Center of Excellence in H2020
  - Centre of Excellence in Simulation of Weather and Climate in Europe

Transitional approach towards the vision

- Scalable data management practice
- **The inhomogeneous storage stack**
- Suboptimal performance & performance portability
- **Workflow awareness**

Community effort: Next Generation Interfaces

Earth-System Data Middleware

University of Reading

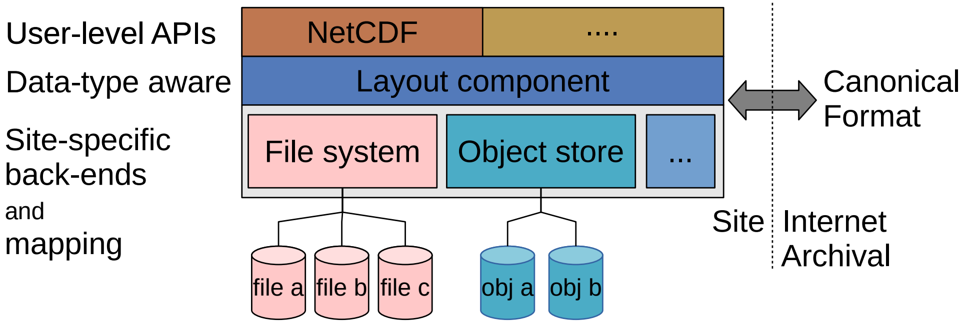
Design Goals of the Earth-System Data Middleware

1. Relaxed access semantics, tailored to scientific data generation
   - Avoid false sharing (of data blocks) in the write-path
   - Understand application data structures and scientific metadata
   - Reduce penalties of **shared** file access
2. Site-specific (optimized) data layout schemes
   - Based on site-configuration and performance model(s)
   - Site-admin/project group defines mapping
   - Flexible mapping of data to multiple storage backends
   - Exploiting backends in the storage landscape
3. Ease of use and deployment particularly configuration
4. Enable a configurable namespace based on scientific metadata

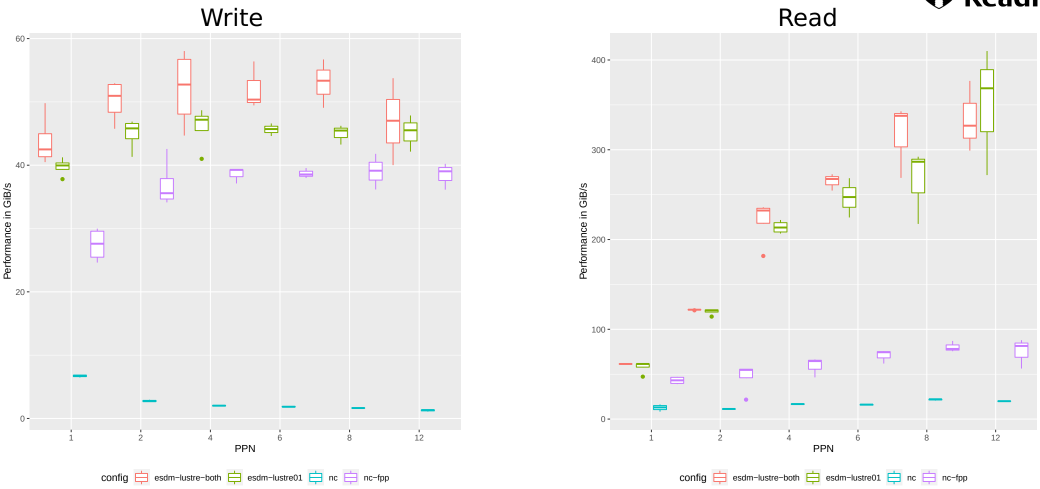
Architecture

Key Concepts

- Middleware utilizes layout component to make placement decisions
- Applications may use existing API (e.g., NetCDF)
- Data is then written/read efficiently; potential for optimization inside library



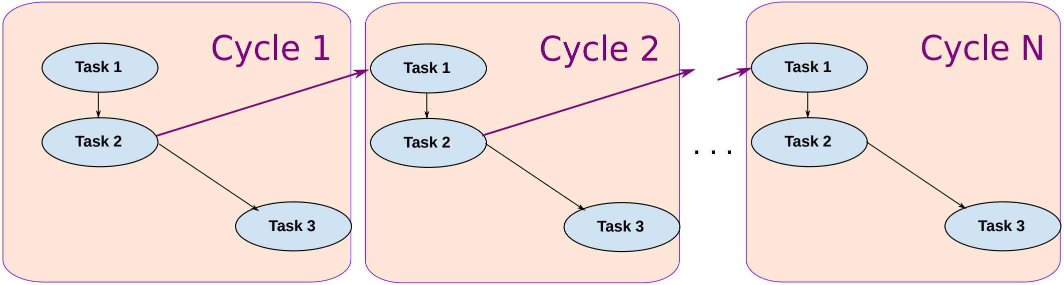
Performance NetCDF-Bench 100 Nodes@Mistral



- Better performance than FPP but looks for users like a single file

Outline

A (Science) Workflow Description



- Current practice (in climate/weather)
- Dependencies between tasks are described
- Assume a calculation that repeats for multiple cycles/iterations

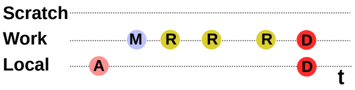
Smarter IO Scheduling: Advantage for Data Placement

University of Reading

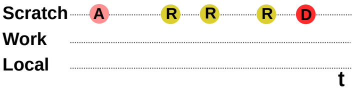
Scenario

- Consider three file systems: local, scratch, and work
  - Local is a compute-node local storage system
- Data can be stored on any of these storage systems
- Users need to manually optimize data placement to hardware throughout life cycle
- Could the system do more knowing details about the workflow?

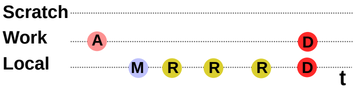
Alternative life cycles for mapping a dataset (Selection)



Local and work file systems



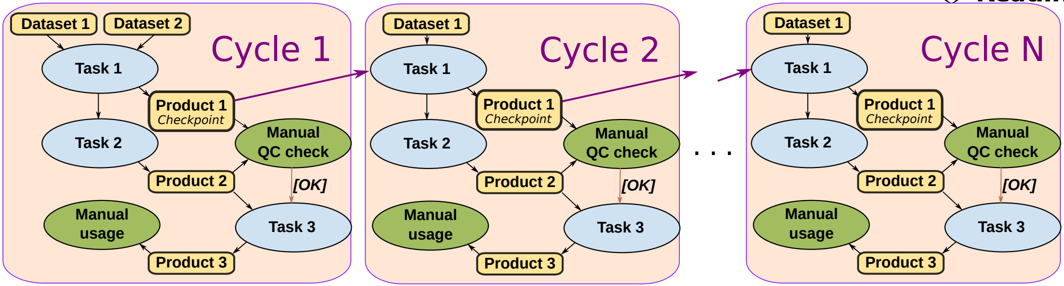
Scratch file system only



Local and work file systems

Allocation, **M**igration, **R**eading, and **D**eleting

Possible Extended (Science) Workflow Description



- Vision: enhance workflow description with IO characteristics
  - Input required
  - Needed input
  - Generated output and its characteristics
  - Information Lifecycle (data life)
  - ⇒ Explicit input/output definition (dependencies) instead of implicit

Planning HPC Resources: An Alternative Universe

- Scientists deliver
  - detailed but abstract workflow orchestration
  - containers with all software
  - data management plan with data lifecycle
  - time constraints and budget
- Data centers and vendors
  - Simulate the execution before workflow is executed
  - Estimate costs, energy consumption
  - Determine if it is the best option to run
- Systems
  - Utilize the information to orchestrate I/O AND computation
  - Make decisions about data location and placement
    e.g., trade compute vs. storage and energy/costs vs. runtime
  - Ensure proper execution
- Provoking: Big data technology is ahead of HPC in such an agenda

Vision: Exploit Workflow Knowledge

University of Reading

- Goal: Providing a separation of concern
  - Scientist declares workflow including IO
  - System maps workflow to hardware using expert knowledge and ML
- Users provide enhanced workflow description
- System performs smarter IO scheduling
  - Considering the hardware/software environment
  - Data placement: Transfer, migration, staging, replication, allocation
  - Data reduction: data compression and data recomputation
- Integration of prototype into ESDM (part of ESiWACE2) ongoing

Outlook: Next Generation Interfaces

University of Reading

Attempt for community building toward an forum

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
  - Utilizing arbitrary compute and storage technology!
- 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

Outline

University of Reading

Identifying Data Properies via Sampling

- Understanding data characteristics is useful
  - Relation of file types to optimize relevant file formats
  - Conducting what-if analysis
    - Influence of compression, deduplication
    - Performance expectations
- Analysing large quantities of data is time consuming and costly
  - Scanning petabytes of data in $>$ 100 millions of files
  - Optimally, with 50 PB of data and 5 GiB/s read, 115 node days (4,000 €)
  - Much slower: experiments for compression paper required 1,600 node years!
  - $\Rightarrow$ Working on a representative data set that reduces costs is mandatory
- Conducting analysis on representative data is difficult
  - What data makes up a representative data set?
  - How can we infer knowledge for all data based on the subset?
    - Based on file numbers (i.e. a typical file is like X)
    - Based on capacity (i.e. 10% of storage capacity is like Y)
  - Many studies simply select a data set and claim it is representative

Contribution

Goal

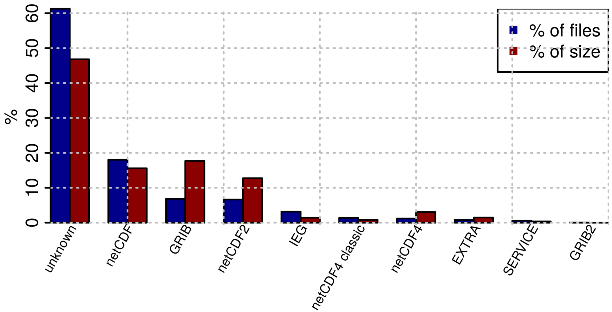
- Investigation of statistical sampling to estimate file properties
  - Can we trust the results?
  - What are typical mistakes when sampling data?
- Conduct a simple study to investigate compression and file types

Approach

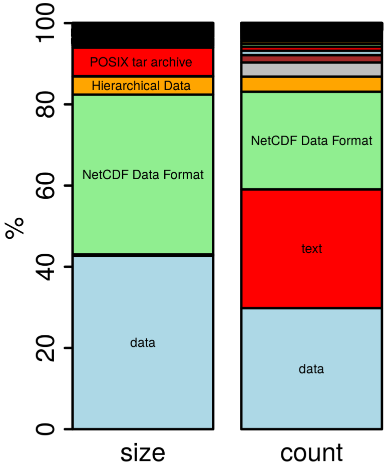
1. Scanning a fraction of data on DKRZ file systems
   - Analyzing file types, compression ratio and speed
2. Investigating characteristics of the data set
3. Statistical simulation of sampling approaches
   - We assume the population (full data set) is the scanned subset
4. Discussion of the estimation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

Scientific File Formats

- The computation by file count and capacity differs
  - The heavy-tailed distribution skews analysis



Type according to CDO tool



Type according to "file" tool

Sampling Strategies

University of Reading

Sampling to Compute by File Count

1. Enumerate all files
2. Create a simple random sample
   - Select a random number of files to analyze without replacement
   - For proportional variables, the number of files can be computed with Cochran's formula
   - You can use simulation to estimate the error for contiguous variables

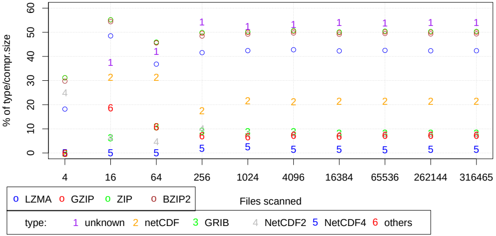
Sampling to Compute by File Size

1. Enumerate all files AND determine their file size
2. Pick a random sample based on the probability $\frac{filesize}{totalsize}$ with replacement
   - Large files are more likely to be chosen (even multiple times)
3. Create a list of unique file names and analyze them (e.g., for a compression study)
4. Compute the arithmetic mean for the variables
   - If a file has been picked multiple times in Step 2., its value is used multiple times

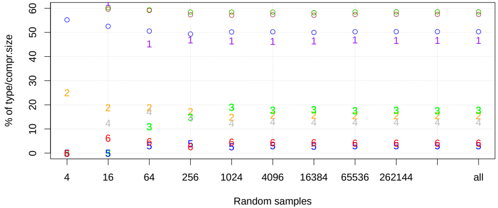
Demonstration of the Strategies

- Apply the approach with an increasing number of samples
  - Compare true value with the estimated value

Running one simulation for increasing sample counts



(a) Compute mean by count



(b) Compute mean by size

Evaluating various metrics (proportions) for an increasing number of samples

- To ensure that convergence is realistic, the experiment had been repeated 100x

Example Study Using Compression on two Systems

University of Reading

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	1132.7	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	4449.1	4602.0

(a) 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	1272.6	3215.6
pithy2011-12-24-3	0.613	1787.5	3535.2
lz4fastr131-17	0.614	1904.8	3610.3

(b) DKRZ data

Table 3: Selected algorithms with good properties (sorted by ratio)

- Developed tool: SFS
- Running 162 algos
- Best algos shown left
- DKRZ: 3 TByte of 50 PB data scanned
  - 5 Weeks, one node
  - LZ4Fast faster than memcpy
- WR: 38.1 GByte of 1.1 TByte scanned

Outline

1 Research Overview

2 Earth-System Data Middleware

3 Workflow Awareness

4 Identifying Data Properties

5 Prediction/Prescribing with ML
- System-Wide Defaults
- Applying Machine Learning

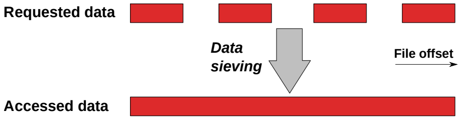
6 Identifying Similar Jobs

7 Summary

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?

Experiments

- 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

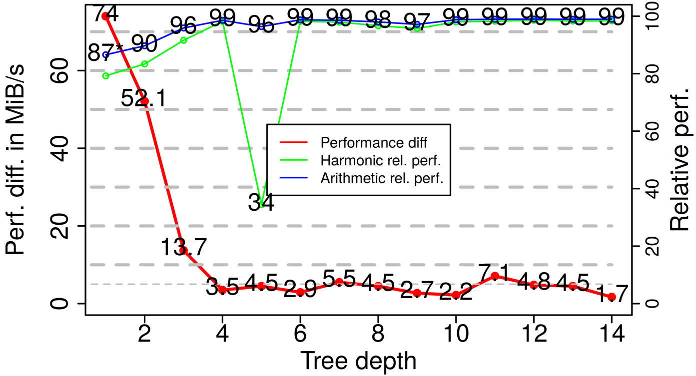
- Comparing a default choice with the best choice
- All default choices achieve 50-70% arithmethic mean performance
- Picking the best default default for stripe count/size: 2 servers, 128 KiB
  - 70% arithmetic mean performance
  - 16% harmonic mean performance ⇒ some bad choices result in very slow 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 classification tree with different depths
- Even small trees are much better than any default
- A tree of depth 4 is nearly optimal; avoids slow cases

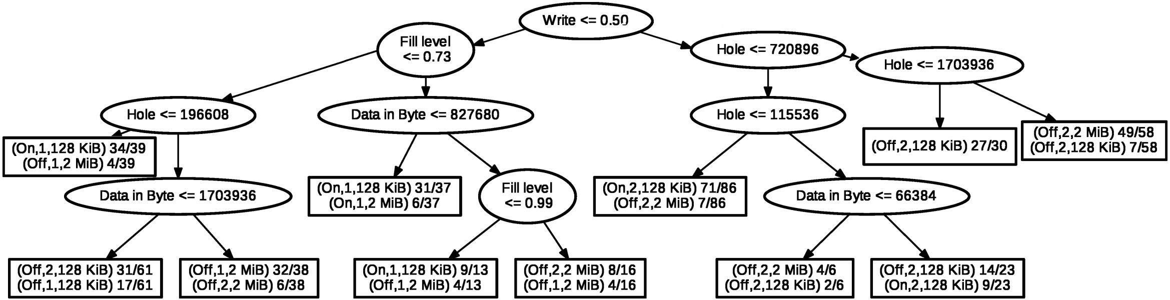


Perf. difference between learned and best choices, by maximum tree depth, for DKRZ's porting system

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

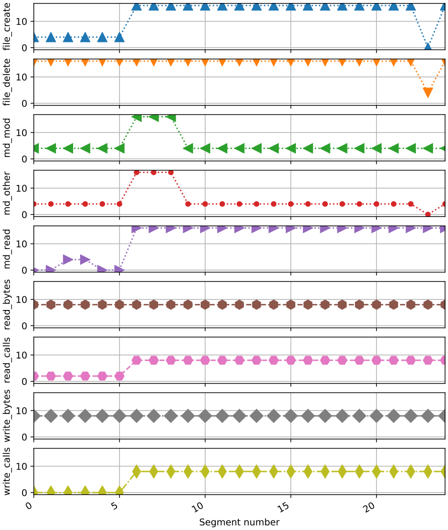
University of Reading

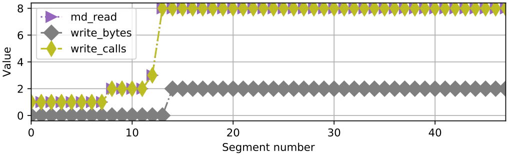
Identifying Similar Jobs Using Time Series

University of Reading

- Problem: 100,000 of jobs are executed on a cluster
  - How can we find similar jobs?
  - Motivation: Estimate benefit of optimization / apply recipies for other applications
- Developed clustering algorithm(s) and workflow for support to investigate jobs
- Derived meaningful distance measures
  - Must compare multiple metrics with different units
  - Must handle variable number of nodes and runtime
- Study on 580,000 jobs (6 months of data from DKRZ)
  - Recorded using DKRZ monitoring system
  - Monitors IO statistics (read/write/metadata) periodically

Two Reference Jobs (Sum for all Nodes)

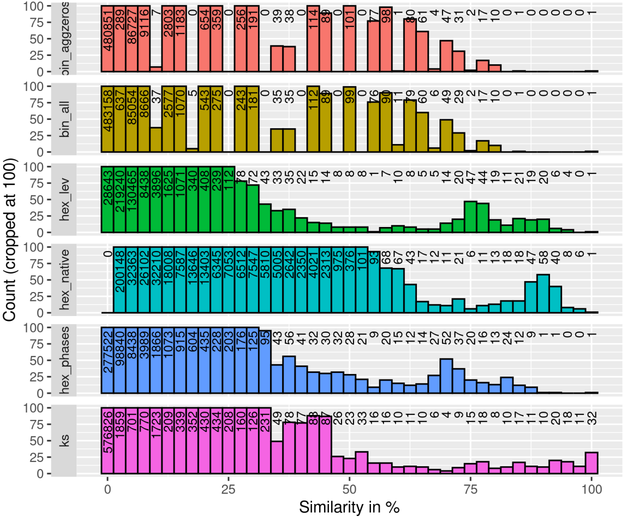




- For this job, other metrics == 0
- Data categorized based on quantiles
  - 0 = non-IO < 99% of all samples
  - 1 = intense < 99.9%
  - 4 = extreme >
- Segments represent 10 min

Similarity to a Post-Processing Job

University of Reading



- Six different algorithms:
  - Aggregation
  - Coding
  - Distance metrics
- User may explore from most similar job to least
  - A cluster is reasonable cut-off
- Quantitative analysis of similarity
  - Users, job names
- Qualitative analysis
  - Inspected 100 most similar jobs

Outline

Summary

- Parallel I/O is complex
  - System complexity and heterogeneity increases significantly
  - ⇒ 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
- To unleash the full potential of system resources, we need:
  - Novel interfaces
  - Workflow knowledge
  - Smarter systems
- I work towards intelligent systems to increase insight and ease the burden for users