Predicting Performance of Non-Contiguous I/O with Machine Learning

Julian M. Kunkel, Michaela Zimmer, Eugen Betke

kunkel@dkrz.de

German Climate Computing Center (DKRZ)

16-07-2015

Outline

About DKRZ

German Climate Computing Center

To provide high performance **computing platforms**, sophisticated and high capacity **data management**, and superior **service** for premium **climate science**.

Scientific Computing

- Research Group of Prof. Ludwig at the University of Hamburg
- Embedded into DKRZ

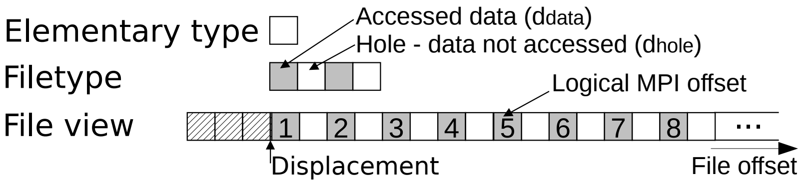
Research

- Analysis of parallel I/O
- I/O & energy tracing tools
- Middleware optimization
- Alternative I/O interfaces
- Data reduction techniques
- Cost & energy efficiency

Motivation

- 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?

Example non-contiguous access pattern in which every other elementary data type is accessed.



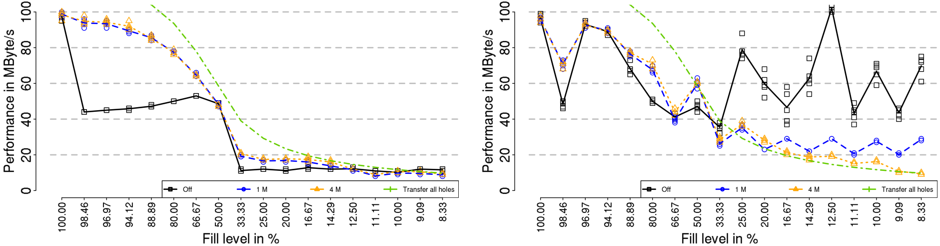
Data Sieving vs. Naive I/O

Limitations of data sieving

- Data sieving accesses data always with the given buffer size

⇒ Smaller buffers may provide better performance

- Data sieving does not work well with complex patterns
- Data sieving does not know anything about file striping



(a) $d_{\text{data}} = 16\,\text{KiB}$ (b) $d_{\text{data}} = 256\,\text{KiB}$

Performance for variable hole size and two block sizes measured with one client.

Goals of the Paper

Goals

- The application of machine learning to determine good settings
- The extraction of rules of thumb (expert knowledge)

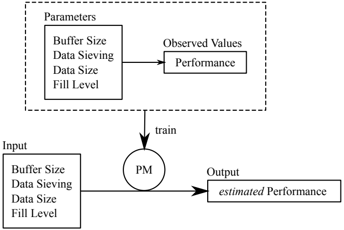
Methodology

1. Measure performance for different patterns
2. Store settings and performance in CSV files
3. Create decision tree (CART) models
4. Evaluate accuracy
5. Investigate training set size
6. Compare benefit over default settings
7. Extract expert knowledge from decision trees

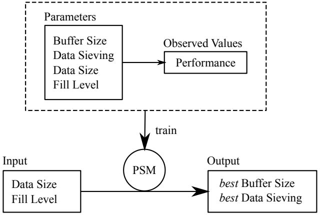
Prediction Models

Alternative models

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



(a) Performance Model



(b) Parameter Setting Model

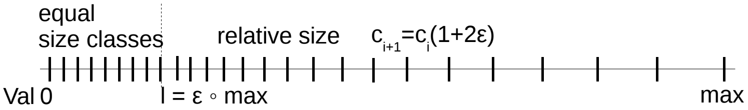
PM provides a perf. estimate, whereas PSM provides the “tunable” variable parameters to achieve it

Transformation of the Problem

- We aim to apply alternative methods from machine learning
- Many require classification problems instead of regression
- Performance values need to be mapped into classes

Mapping

- Create 10 classes with the same length up to 5% of max. perf.
- Then increase performance range covered by 10% each



Evaluation Data

We analyzed the validity of the approach on two systems

System 1: WR cluster

- Lustre 2.5
- 10 server nodes
- 1 Gb Ethernet
- 1 client node (max performance 110 MiB/s)

System 2: DKRZ porting system

- Lustre 2.5 provided by Seagate ClusterStor 9000
- 2 servers
- FDR-Infiniband
- 1 client node (max performance 800 MiB/s)

Validation on Data of the WR Cluster

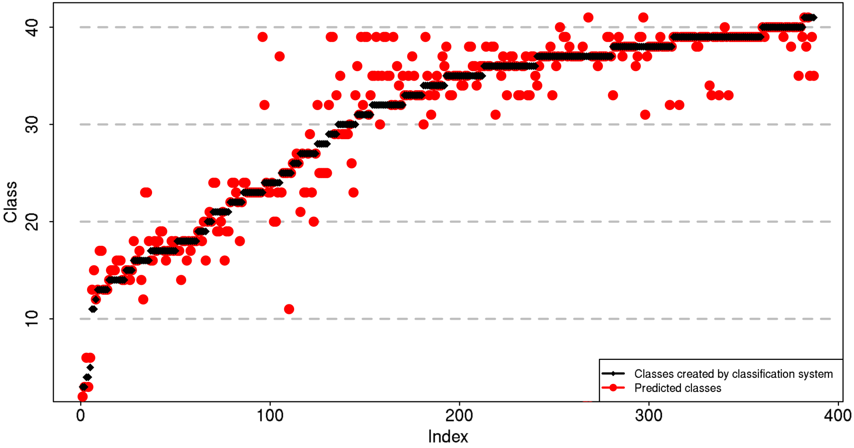
- Apply k-fold cross-validation
  - Split data into training set and validation set
  - Train data with all (k-1) folds and evaluate model on 1 fold
- A baseline model is the mean performance (54.7 MiB/s)
  - Arithmethic mean error is 28.5 MiB/s
- Linear models yield a mean error of $\geq$ 12.7 MiB/s

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. Values for k=3..7 lie in between

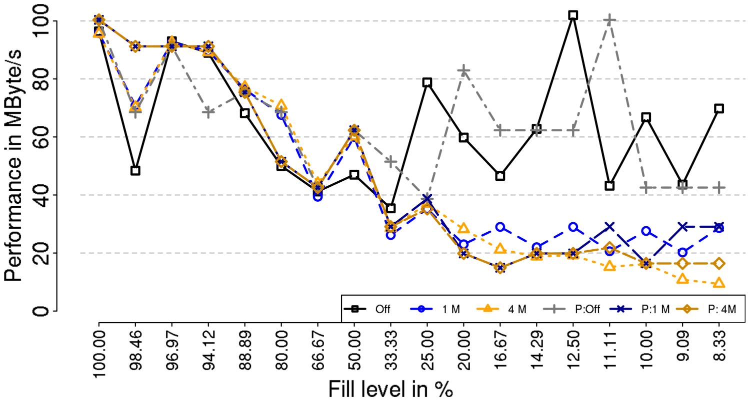
Comparing Prediction with Observation



Sorted CART prediction (trained by 387 instances)

Comparing Prediction with Observation

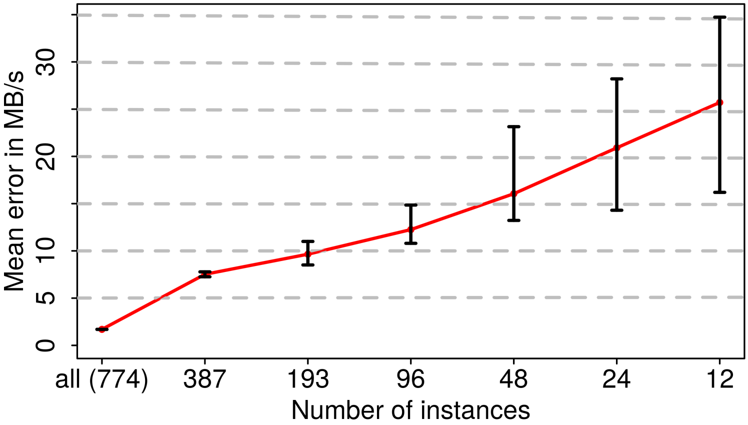
- Non-linear performance behavior causes errors
- Mispredictions due to sparse training data



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

Investigating Training Set Size

- Inverse k-fold validation: learn from 1 fold and test on (k-1)
- With $\geq$ 96 instances better than the linear model



Mean prediction error of PM by training set size under inverse *k*-fold cross-validation. Class prediction errors

Machine Learning vs. System-Wide Defaults

Performance gain over fixed default parameters

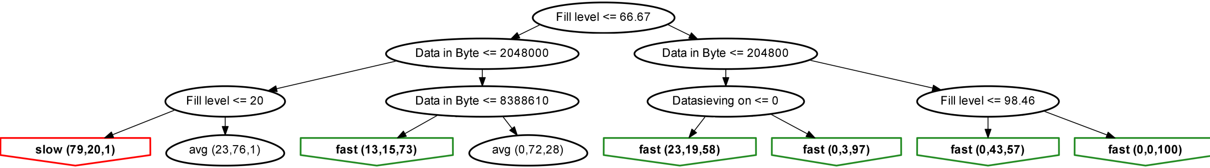
- Best default choice would be data sieving with 1 MiB
- Benefit of CART vs. default is between 25-50%

Default Choice	CART PSM, 387 Inst.	Loss compared to best choice
Off	4.2 MB/s	9.6 MB/s
1 MiB	1.9 MB/s	7.6 MB/s
4 MiB	6.9 MB/s	12.2 MB/s
100 MiB	6.9 MB/s	12.2 MB/s

Arith. mean perf. improvements with the PSM-learned and best choices for s_{buffer} compared to a default

Extracting Knowledge

- Rules can be easily extracted from decision trees
- Consider a performance prediction
- Rules (this is common sense for I/O experts)
  - Small fill levels and data sizes are slow
  - Large fill levels achieve good performance



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.

Measured Data

- Captured on DKRZ porting system for Mistral
  - Evaluate if machine learning could be useful for our next system
- What Lustre and data sieving settings are useful defaults?
- Vary lustre stripe settings
  - 128 KiB or 2 MiB
  - 1 stripe or 2 stripes
- Vary data sieving
  - Off or 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

Comparing Advantage of Settings

- With each setting a few cases achieve better performance
- Turning data sieving on is better in 50% of the cases
- Turning data sieving off is better in 12.5% of the cases
- With DS: 2 and 1 servers in 25% better than 1 and 2

Data sieving			Off				On			
Server count			1		2		1		2	
Stripe size			128K	2 M	128K	2 M	128K	2 M	128K	2 M
Sieving	Server #	Stripe								
Off	1	128 KiB	-	5	37	1	32	32	47	46
		2 MiB	2	-	35	1	32	33	54	47
	2	128 KiB	61	64	-	9	43	44	36	39
		2 MiB	64	64	45	-	46	50	58	54
On	1	128 KiB	125	126	132	108	-	29	76	51
		2 MiB	115	115	122	96	1	-	70	36
	2	128 KiB	114	114	118	109	73	74	-	47
		2 MiB	119	118	114	109	69	69	9	-

Frequency in which a setting of the row is better by 10% (at least 5 MB/s) than that shown in the columns, out of 240 hole/size configurations.

System-Wide Defaults

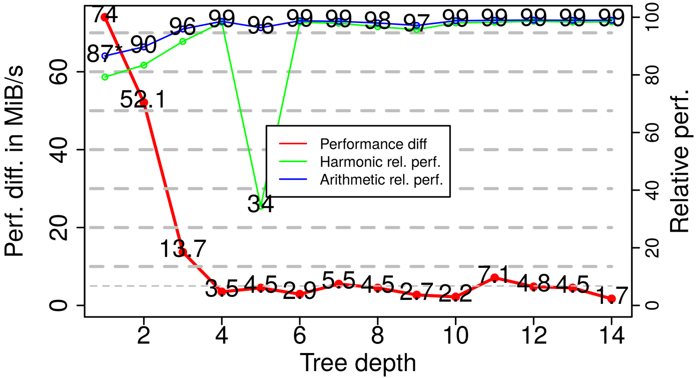
- All choices achieve 50-70% arith. mean perf.
- Picking the best default default choice: 2 servers, 128 KiB
  - 70% arithmetic mean performance
  - 16% harmonic mean performance

Default Choice			Best	Worst	Arithmetic 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

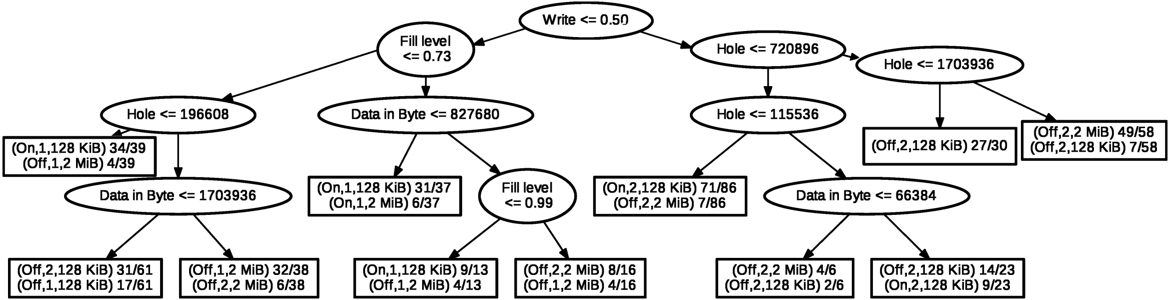


Perf. difference between learned and best choices, by maximum tree depth, for DKRZ's porting system

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 choice



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

Summary

Conclusions

- Non-contiguous I/O optimization is non-trivial to parameterize
- Machine learning is helpful to extract useful I/O settings

$\Rightarrow$ Expert knowledge can be verified or gained

- Even small trees achieve much better results than best default

Ongoing and future work

- Analyse performance of (now) deployed full system
- Provide an optimized non-contiguous algorithm
- On-line assessment of observed performance
- *Please see our poster*