DOI: 10.14529/jsfi160303

Predicting I/O Performance in HPC Using Artificial Neural Networks

Jan Fabian Schmid[1], *Julian M. Kunkel*[2]

The prediction of file access times is an important part for the modeling of supercomputer's storage systems. These models can be used to develop analysis tools which support the users at integrating efficient I/O behavior.

In this paper, we analyze and predict the access times of a Lustre file system from the client perspective. For this purpose, we measured file access times in various test series and develop different models for predicting access times. The evaluation shows that in models utilizing artificial neural networks the average prediciton error is about 30% smaller than in linear models. One phenomenon in the distribution of file access times is of particular interest: File accesses with identical parameters show several typical access times.

The typical access times usually differ by orders of magnitude and can be explained with a different processing of the file accesses in the storage system – an alternative I/O path.

We investigate a method to automatically determine the alternative I/O path and quantify the significance of knowledge about the internal processing. It is shown that the prediction error is improved significantly with this approach.

Keywords: file system, performance, modeling I/O, artificial neural networks.

Introduction

Tools are demanded that help users of HPC-facilities to implement efficient Input/Output (I/O) in their programs. It is difficult to find the best access parameters and patterns due to the complexity of parallel storage systems. The processing of file accesses in a storage system can be viewed as a task that is sequentially propagated along an I/O path in the storage system. Starting at the invoking processor, the storage system is searching for the data, going further and further through the memory hierarchy, until all data is found, so it can be returned to the processor.

Currently, users have to optimize their programs for each system individually without much assistance. To develop tools which support the implementation of efficient I/O, computational models of the storage system are important. For instance, a tool could estimate the I/O path and classify accesses as outliers if they behave abnormally. For single hard disk systems such a model can be derived analytically [11]; however, for the complex storage system of a supercomputer, these models become too difficult to configure [13].

In this paper, we evaluate predictors of I/O performance using machine learning with artificial neural networks (ANNs). In our analysis, we use ANNs with different input information for the prediction of access times. Additionally, we evaluate strategies to identify the I/O path without a-priori (expert) knowledge of it.

Because of the strong linear correlation between access time and access size, the problem seems to fit linear models. However, our results show that the relation of file access parameters to access time is not sufficiently represented by linear models. ANNs achieve significantly better results than linear models. Our analysis suggests that the I/O path used by the storage system considerably influences the file access time. Therefore, it becomes key for a good model of access

[1]Universität Hamburg, Hamburg, Germany; E-Mail:`2schmid@informatik.uni-hamburg.de`
[2]German Climate Computing Center (DKRZ), Hamburg, Germany

times to derive knowledge about I/O paths. Unfortunately, I/O paths are difficult to deal with, as it is unknown which path was used for a file access.

This paper is organized as follows: Related work is provided in Section 1. In Section 2, we explain our analysis of file access times. Firstly, we develop a simplified model of the I/O path. Next, a method for approximating I/O paths with derived classes from the error of a prediction model is proposed. At last, we introduce a set of models for access time prediction. Afterwards, in Section 3, we evaluate our analysis of the storage system by measuring access times in various test series, studying the obtained access times and the predictions of our models to them. In the end, the final Section summarizes the paper and suggests future work.

1. Related work

Generally, storage systems are modeled in two different ways for access time prediction; either using white-box modeling or black-box modeling [4].

- **White-box modeling**: The storage system itself is simulated. Details of hardware components like rotation speed of the magnetic disk in a hard drive are considered. The processing of a file access can then be simulated in the model and the resulting access time is then used as prediction for the actual system. Processing and resulting performance can be analyzed in detail on the model.
- **Black-box modeling**: The model abstracts from the real storage system. System performance is approximated without considering the causes. This procedure corresponds to an emulation of a storage system. In contrast to the white-box model, processing of file accesses can't be analyzed on the model itself.

These two ways of modeling are fundamentally different and have to be differentiated.

1.1. White-box modeling versus black-box modeling

For the in-depth analysis of reasoning for behavior of a storage system, a white-box-model is desirable. On the one hand, the modeled system is represented in the model. Thus it, can be examined. On the other hand, these models can be very precise if modeled correctly [11]. There are important limitations of white-box modeling, however. For every system an individual model has to be created and a model becomes quite intricate for single hard drives already [4]. To approach the complexity of white-box modeling, Ruemmler and Wilkes analyzed the relevance of different hard drive components for the model deviation to save effort for insignificant parts [11]. However, white-box modeling is usually used for simple systems like a single hard drive. For these hard drives white-box-modeling is already very demanding, hence for the complex parallel storage system of a supercomputer it's not a feasible approach [13].

The application of black-box-modeling is easier and more flexible as it's independent from the individual system. Stochastic approaches coupled with data mining methods are mostly used for black-box modeling; for example, a combination of regression trees can be used [6], or selective bagging classification and regression trees [13].

1.2. Prediction of I/O performance with ANNs

Computability of ANNs was researched by Rojas [10] and Cybenko [5]; they demonstrated the possibility of modeling non-linear systems. Cybenko also proved the *universal approximation* *theorem*, which states that feed-forward networks with sufficient complexity can approximate any continuous functions on compact subsets of $\mathbb{R}^n$. Therefore, ANNs should be sufficient for predicting I/O performance as long as there are no contradictions in behavior. However, no statement can be made about the necessary structure of the network for this task.

Crume et al. developed a method using ANNs which exploits periodic patterns in sequences of file access times [4]. A Fourier analysis is used to determine the most important frequencies, which are then used as input information for the ANNs. In a following publication [3] they move away from Fourier analysis, but instead use additional sine waves as input for the ANNs. This seems to be a promising approach for predicting access times of single hard drives, where the rotational characteristics of the magnetic disk are an essential consideration. It is, however, difficult to extrapolate behavior from a single disk to a distributed storage system consisting of thousands of disks that utilizes several optimization mechanisms.

1.3. I/O performance prediction in HPC

Performance analysis in HPC is an important task to examine and improve system efficiency. For instance Liu et al. simulated scheduling algorithms for research [9]. The simulation used the white-box modeling tool DiskSim for a prediction of occurring file accesses [1].

There are a few simulators for parallel storage systems, for example CODES [2] utilizes a scalable infrastructure to investigate relevant research issues, such as the importance of burst-buffers. PIOSimHD is a simulator that is able to replay (MPI) application traces on a generic I/O system [8].

Analytical and machine learning models for predicting performance are another choice to optimize performance. Kunkel et al. utilized access time prediction with decision trees for varying parameterizations of ROMIO for access of non-contiguous data [7]. Instead of searching for optimal parameters by testing, they were able to find good values through estimating performance.

2. Modeling I/O Access Times

2.1. Characteristics of the Data

We used a synthetic benchmark in which identical measurements for random or sequential I/O are performed and the access time is measured for each operation individually. The resulting timings are stored together with file access parameters in a file that is then used to build and validate the models. This approach allows for an in-depth analysis of system behavior for different use cases from the perspective of a single client. The stored parameters are:

- **Access size**: Number of bytes to read or write.
- **Access type**: Differentiates reading and writing.

Directly measurable or derivable attributes are:

- **Offset**: Distance of file beginning to the starting point of access.
- **Delta-Offset**: Can be calculated for file access i as Offset[i] - (Offset[i-1] + access size[i-1]).
- **Access time**: Time for performing the I/O.

Knowledge of internal aspects of the system about the current system utilization or about the storage media that have to be addressed are not used by the model.

2.2. Model of the I/O path

The internal processing of a file access in the storage system can be viewed using the I/O path. The I/O path of a file access is the interaction between all components that are involved in the transfer of data read or written for its execution. For example, for a node local file system this includes operating system, memory and storage device/medium. Remote systems add network traffic and generally server sided components. Usually, in parallel file systems the I/O of the client is transferred through operating system, client side file system modules, network, file system servers, storage devices and ultimately media. In several of these steps, data may be cached in memory and optimizations can take place.

The resulting access time depends on the depth of this I/O path, because with increasing distance the storage media along the path is slower. While the first levels in memory hierarchy (CPU caches) are the fastest volatile storage, the main memory is already an order of magnitude slower; the same applies for the step into the parallel storage system, that is deployed as servers connected via network to the computer nodes. Reading file accesses are more severely affected by varying depths of I/O paths than writing file accesses, which can be deferred using write-behind and propagated lazily to the disk drives.

2.2.1. I/O path for access time prediction

Due to the exponential decay of processing speed in the hierarchy, file accesses with similar access parameters, but varying I/O path, can be easily differentiated by a step in the magnitude of access time. As the access time is dominated by the slowest component along the I/O path, a step in the measured access time occurs between the I/O paths of diverging length. In Figure 1, measurements of sequential read accesses are shown. The different groups of measurements with equal access parameters (points with the same color) are clearly visible. For a few access sizes, some clusters are observable, this step in the magnitude of access time and can be explained with varying I/O paths (this figure will be discussed in further detail in the following chapters).

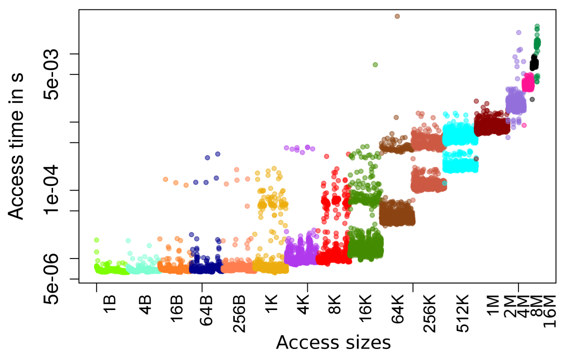

Figure 1. Time series for sequential reads; access sizes increase from left to right and all measurements with equal access parameter values have the same color

2.3. Estimating the I/O path using error classes

We utilize a concept that we call *error classes*. Each class is supposed to approximate an I/O path based on the error (residuum) of the observed access times to a baseline model. The baseline model is constructed as a function of the file access parameters.

At a first consideration, one could come up with the idea of simply clustering the data seen in Fig. 1, because the groups of measurements with varying I/O paths might be correctly differentiated by it. However, it is not a good approach. One would have to cluster for every set of measurements with identical access parameters individually, as access time is strongly dependent on the access size. In a real application scenario, a large number of different access parameters with only very few instances occur, which makes this approach unfeasible.

Another idea might be to use the difference of an arbitrary value to the measured file access time as a parameter for the clustering. Such a value would not be appropriate for all magnitudes of access time occurring in the data. I/O paths with only a small difference in their typical access time might not be found with this approach, because the gap between them doesn't matter at the scale of the chosen value.

Our approach uses the residues of a model like linear regression that can't differentiate between measurements with equal access parameter values. Basically, the linear regression predicts an average access time for the measurements of any given size. The deviation of an individual measurement for these access parameters to the predicted average value is then characteristic for the I/O path. Clustering the residues with a k-Means algorithm allows us then to assign each measurement to a specific cluster, which is the approximation of its I/O path. We call the retrieved clusters **error classes**. We use the absolute error of linear regression as a clustering parameter, so that one cluster can, for example, represent a positive deviation of 1 millisecond. Ideally, it might be the case that file accesses independent of the access parameters that took 1 millisecond longer than usual are processed on the same I/O path. In that case error classes are directly corresponding to I/O paths. However, this has to be investigated.

We use error classes to quantify the importance of knowledge about I/O paths for access time prediction. For the actual prediction of access times, error classes can't be used, because a measured access time is required to determine the error class of measurement. However, the method could be used for a tool, which analyses the execution time of I/O on the client side made during application of a program and assign the observed time to the error class and, thus, the I/O path. It could analyze whether file accesses were slower than expected and therefore unfavorable I/O paths were used during their execution. If a direct correlation of error class to I/O paths were found, such a tool might be able to inform about percentages of used I/O paths.

2.4. Models

Access times of file accesses can be predicted using various models. While we evaluated several models for this paper, only a few relevant models are described; more models are described in [12]. Each model seeks to correlate its known file access parameters to the corresponding access time. Linear regression is used as a baseline model with a simple mapping of access size to access time. Additionally, three models with different input information utilizing ANNs are used:

- The most simple ANN-model only receives information about access sizes, delta-offsets and access types as input.
- The second ANN-model is called **ema-model**. Additionally to the parameters of the previous model, it receives information about the past data throughputs of the system. This can be used to exploit time dependencies of the I/O performance.
- The third ANN-model, called **error-class-model**, receives error classes in addition to the access parameters of the first ANN-model.

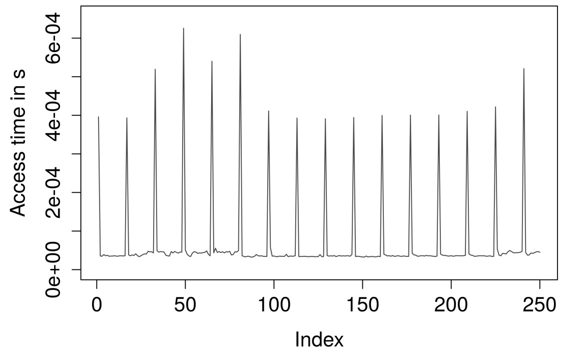

Figure 2. Small sequence of read accesses with periodic peaks in access time

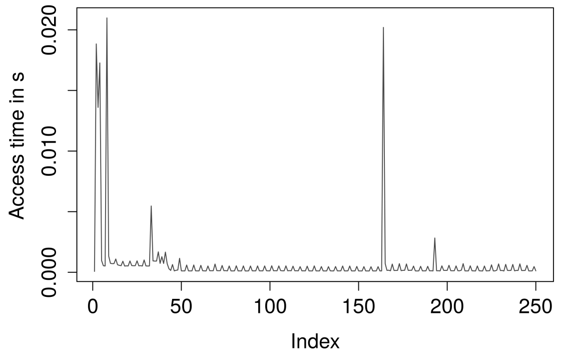

Figure 3. Small sequence of read accesses with sporadic peaks in access time

The error-class-model is used as described in Section 2.3 to examine the potential improvements for access time prediction due to knowledge about I/O paths. Therefore, we evaluate how much the prediction of access times of a model using error classes improves compared to a model without.

The idea of the ema-model is to exploit periodic changes in the performance of the storage system. Our analysis of measurements as well as the work of Crume et al. [4] indicate that periodic phenomena of access times in sequences of files accesses can be exploited for access time prediction. Figure 2 shows a small series of measurements of sequentially read file accesses. The peaks in access time are occurring periodically in this particular sequence. Such behavior appears only partially, often access times fluctuate without observable periodicity as in Figure 3. An occurrence of a periodic pattern could be explained with a processing of the storage system in which a larger amount of data is loaded into the cache at the same time as soon as a cache-miss happens.

With knowledge about the periods of peaks a model could make better access time predictions. The ema-model is supposed to use its input information of previous data throughput for that. The data throughput has peaks in the same period as the access time. We use the data throughput instead of the access times of measurements, because an analysis of the data throughput could find periodic behavior of system performance even in a sequence with different access sizes. Internally the model calculates the exponential moving average (EMA) of data throughput of measurement i as:

$$EMA(i) = 0.5 \cdot t(i) + 0.5 \cdot EMA(i-1) \tag{1}$$

With $t(i)$ the access time of measurement i. The influence of data throughput of a measurement is exponentially decreasing in the series of EMAs. In Figure 4 the EMA-function for a function with periodic peaks can be seen. The ema-model can use $EMA(n-1)$ for the prediction of access time of measurement n. If the model is able to find threshold values of the EMA-function after which another peak follow, it can use this input information for better access time predictions on sequences with periodic peaks in access time. Depending on how relevant periodic behavior of access times is in our measurements, this additional input information could improve the model.

Additional models that were examined in [12], but won't be analyzed in further detail in this paper, are summarized in the following list:

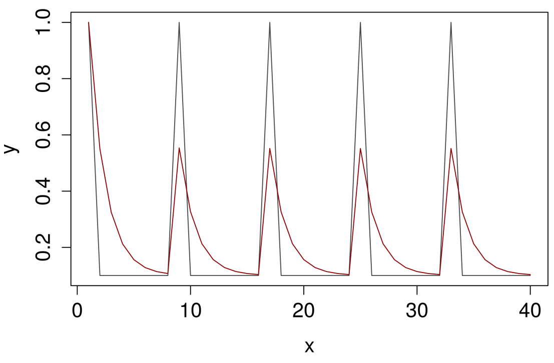

Figure 4. Exponential moving average (in red) for a function with periodic peaks

- Using multiple linear regression, we analyzed the performance of other baseline models. We used access size and delta-offset as tuple and also access size, delta-offset and access type as triple to find a mapping to access time. Despite having more information to work with both models achieve equal or worse deviations to the measured access times than the simple linear regression which only considers access size. If access type and delta-offset have a meaningful correlation to the access time of a file access, they require more sophisticated tools than linear regression to be exploited.
- An ANN-model receiving knowledge about the previous measurement was examined as well. Additionally to the input information of the simple ANN-model the access size, the access type, the delta-offset and also the access time of the previous file access was given as input for this model. Conceptually this model might be able to compare the actual performance on the previous file access with its own prediction for it. If the actual performance deviates from the predicted performance this model could exploit this knowledge to adapt its prediction for the following file access. In case of a prolonged period of high workload this adaptation might have positive effect on the predictions. The results of this model, however, were throughout worse compared to the simple ANN-model which has less information to work with, even though quite complex network structures were used in the learning process (16 hidden layers with 17 neurons each for the sequential access pattern and 14 hidden layers with 17 neurons each for the random access pattern achieved the best results). Therefore we conclude that the performance of the direct predecessor of a file access can't or is too complicated to be exploited for an improved prediction.
- An ANN-model similar to the error-class-model described above, using error classes obtained from the residues of the simple ANN-model instead of residues from the linear regression, was examined as well. This approach lead to very similar prediction performance to the model using error classes from linear regression. On the data set of sequential file accesses this model achieved with $2.4 \cdot 10^{-5}$ to $2.0 \cdot 10^{-5}$ a slightly worse and on measurements with random access pattern with $8.9 \cdot 10^{-4}$ to $10.3 \cdot 10^{-4}$ a slightly better mean average error.

3. Evaluation

The execution and results of analysis are summarized in the following way. First, the test system used for measurements and the strategy for systematic measurement are presented. Next, we explain the computation of error classes. Finally, the quality of predictions of the different models are examined.

3.1. Test system

The measurements were done on the supercomputer Mistral (phase 1 system) of the DKRZ (Deutsches Klimarechenzentrum)[3]. The phase 1 system of Mistral operates with the parallel distributed file system Lustre (Lustre 2.5, Seagate's edition) and consists of over 1500 compute nodes and 30 Petabyte storage. Each compute node consists of two E5-2680v3 with a clock rate of 2.5 GHz and 30 MiB L3 Cache. Measurements were done during daily operation, typical fluctuations in workload may have influenced the execution of benchmarks.

3.2. Benchmarking

We created a simple benchmark (io-model) that used POSIX read/write() and measured time for each I/O individually. In this paper, it is executed on one node with one processor using a single stripe (one OST) to study the quality of the predictions. Each series of measurements follows a certain pattern to study system behavior systematically.

Sequential and random I/O patterns are measured, each with reading and writing file accesses. Our test file has a size of 10 GiB which theoretically fits in the main memory of computer nodes, but measurements and observations from Lustre's /proc statistics suggest that caching is not effective for this configuration (sequential reads are improved by read-ahead, though). Occasional access of hard drives in the parallel storage system is likely. Most series of measurements contain 10 000 file accesses with a certain access size, only series with sequential access and an access size of greater than 2 MiB have fewer measurements as the end of our 10 GiB file is reached. Every series with specific access pattern, access size and access type is repeated three times. Access sizes are varying from 1 B to 16 MiB (in detail: 1 B, 4 B, 16 B, 64 B, 256 B, 1 KiB, 4 KiB, 8 KiB, 16 KiB, 64 KiB, 256 KiB, 512 KiB, 1 MiB, 2 MiB, 4 MiB, 8 MiB and 16 MiB).

3.3. Analysis of measurements

The measured access times for the four different cases are summarized in Table 1.

Table 1. Overview of file access times for the different cases

Case	Min. value	1. Quartile	Median	Arith. mean	3. Quartile	Max. value
Sequential reading	6.2e-06	6.9e-06	9.7e-06	3.8e-04	1.3e-04	5.6e-02
Sequential writing	7.5e-06	8.4e-06	1.6e-05	4.3e-04	2.2e-04	3.0e-02
Random reading	6.5e-06	1.4e-05	1.8e-03	7.4e-03	1.3e-02	5.3e-01
Random writing	1.1e-05	2.6e-04	4.5e-04	3.8e-03	2.2e-03	2.1e+00

As expected, sequential file accesses are on average much faster than random accesses especially for the reading case, also random reads are slower than writes. In Table 2, we consider the correlations between file access parameters and the resulting access time, a strong correlation can be exploited for access time prediction.

Table 2. Correlations between file access parameters and access time

Attribute	Sequential access pattern	Random access pattern
Access size	0.973	0.3291
Delta-offset	NA	0.0069
Access type	0.018	-0.1068

[3]`http://www.vi4io.org/hpsl/2016/de/dkrz/start`

Note that sequential access leads to a constant delta-offset of 0.
Sequential file accesses have with 97.3% a clear correlation to access time, this is with only 32.91% to a lesser degree still true for the random accesses. This is why linear regression can be expected to have acceptable results for access time prediction, especially for the sequential access case.

To study the distribution of measured file access times we display them as follows: All measurements with equal access size have the same color, access sizes of file accesses are increasing in the graphs from left to right. The resulting graphs can be seen in the Figures 5 to 8. Note that they use logarithmic scale.

For readability, the slowest and fastest 1% of measurements are purged and only every 25th measurement is plotted in the figures.
The correlation between access size and access time can be easily identified. As described in chapter 2.2.1 the split of measurements with identical access parameter values, that occurs in particular in the cases of sequential file accesses, can be explained with a processing in the storage system along different I/O paths. It is worth mentioning that the groups of access time for different access sizes are partially at the same range of magnitude. This indicates as well that the splits are caused by varying I/O paths as one I/O path has a typical access time, which only changes slightly for different access sizes.

The phenomenon of these splits clarifies why despite the strong correlation of access size to access time a linear model might produce sub-optimal predictions. Even for identical access parameter, it is difficult to predict which I/O path will be used as the storage system decides the I/O path based on the system state (e.g., is the data available in the client's cache). In Figure 7, the importance of knowledge about I/O paths for the access time prediction can be seen: For example, file accesses of only 1 B can be processed as slow as accesses with 16 MiB if the longest

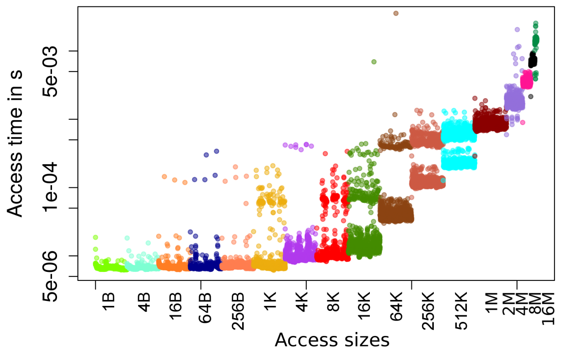

Figure 5. Measurements of sequential reads

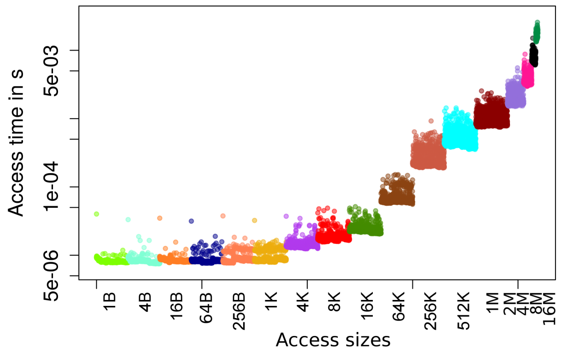

Figure 6. Measurements of sequential writes

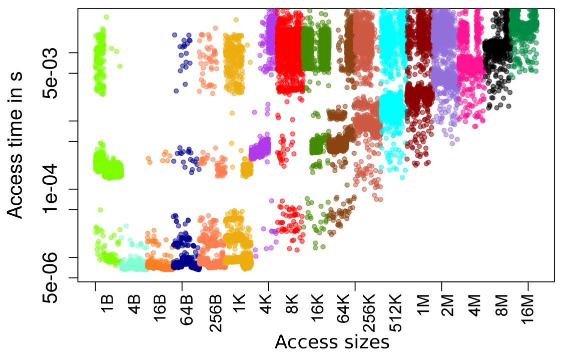

Figure 7. Measurements of random reads

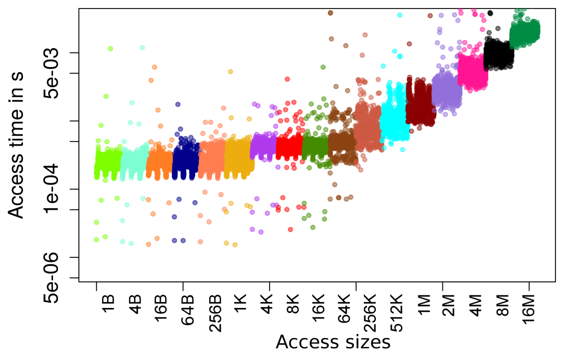

Figure 8. Measurements of random writes

I/O path is used. For writing file accesses the splits only occur occasionally and they are less drastic. These measurements might be well represented with a linear model.

3.4. Analysis of error classes

Error classes are calculated as clusters applying the k-means algorithm on the residues of the linear regression model. As an estimation of possible I/O paths we classify into 10 clusters. In Figure 9, the predictions of linear regression on the random reads can be seen as blue points. Since a linear model predicts the mean value for all measurements with identical arguments, it cannot distinguish between the different I/O paths and may not even predict a valid value (e.g., the median).

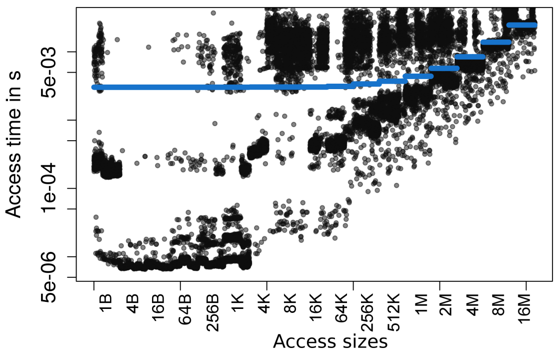

Figure 9. Predicted access times of the linear regression model as blue points

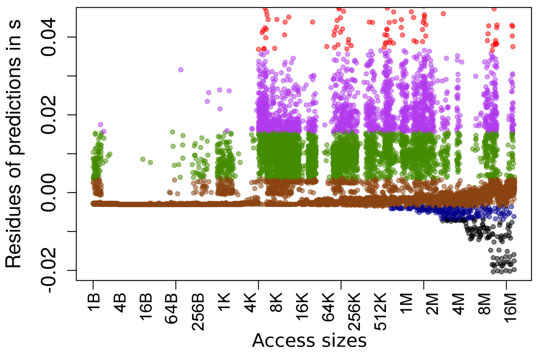

Figure 10. Residues of the linear regression model, colors are corresponding to error classes

For identifying the error classes, the difference between the prediction of linear regression and actual access time is used as input for the clustering algorithm. In Figure 10, the residues for all measurements are shown. The different colors correspond to the computed clusters, which are our error classes. Note that due to the random initialization of k-Means, different runs may result in different clusters [4].

The clusters are differentiated along horizontal lines in the graphs. Without going into further details, one can recognize from the graphs that the error classes are not perfectly approximating the I/O paths. The measurements with the smallest access sizes (on the far left) are only differentiated into two different error classes. However, the two error classes mainly represent measurements with positive errors, and close to 0 or negative error, what seems to be a reasonable approximation of I/O paths.

The computed error classes can be examined in further detail with the data from Table 3. Ideally for a good correlation of I/O paths to error classes, each error class should represent one particular value of data throughput that would be characteristic for the slowest storage medium occurring in the I/O path. This is mostly true for the error classes. The classes 4, 5 and 6, however, have with $5.5 \cdot 10^7$, $6.1 \cdot 10^7$ and $5.8 \cdot 10^7$ very similar average throughput values, which might be a hint for overfitting error classes.

The vast majority of measurements were assigned to error class 3, which is the error class that covers the range around a prediction error of 0. Therefore, most access times can be predicted sufficiently by the linear regression model.

[4]We will consider more robust algorithms in the future.

Table 3. Characteristics of computed error classes for random file accesses in detail

	averaged values			prediction error			
Error class	Throughput (B/s)	Access size (B)	Access time (s)	Min (s)	Avg (s)	Max (s)	# of members
1	1.4e+09	1.5e+07	0.0130	-0.0210	-0.0090	-0.0069	9467
2	9.9e+08	9.1e+06	0.0101	-0.0069	-0.0047	-0.0036	54371
3	2.3e+08	1.4e+06	0.0024	-0.0036	-0.0025	0.0036	825974
4	5.5e+07	1.2e+06	0.0143	0.0036	0.0096	0.0156	85462
5	6.1e+07	2.3e+06	0.0276	0.0156	0.0216	0.0366	37862
6	5.8e+07	4.0e+06	0.0598	0.0366	0.0516	0.0976	4695
7	3.2e+07	4.7e+06	0.1528	0.0977	0.1438	0.2066	1443
8	4.5e+06	1.2e+06	0.2741	0.2067	0.2696	0.4728	567
9	3.0e+05	1.6e+05	0.6956	0.4822	0.6923	1.0063	123
10	9.4e+02	1.0e+03	1.3627	1.0396	1.3597	2.1216	36

One sign for correct approximation of I/O paths are error classes which have equal access sizes but different access times. In such cases a differentiation of measurements with equal access parameters takes place. This occurs for example with the error classes 4 and 8. Both classes have an average access size of $1.2 \cdot 10^6$ Bytes, but with 0.0143 and 0.2741 seconds very different average access times. The error classes of sequential file accesses display similar characteristics as the analyzed random file accesses. However, since sequential accesses are not able to stress all different I/O paths, this result is more interesting.

3.5. Prediction of file accesses

Next, we study the results of our models. To investigate overfitting and the accuracy of the models, we split the available data into a training set and a test set. The test set consists of all measurements that weren't used for training. The parameters for the ANN-models are varied to explore appropriate configurations; they are trained with a wide span of different parameter values for the number of hidden layers and the number of neurons per layer. They had 1000 randomly chosen measurements for each pair of access size and access type as training set. In total, that are 34 000 measurements which should be a significant amount of data to avoid the situation, in which too few data points are available to train the system.

In Tables 4 and 5 characteristics of the ANNs with the smallest average prediction error can be seen. The models for the sequential data set had in general bigger network structures. We do not completely understand the reason, as we naively expect that random I/O is more complex than sequential I/O. Presumably, the system uses additional layers to adjust for rare events or, for example, to differentiate read-ahead and write behind cases.

Table 4. Characteristics of the most successful ANN-models for seq. file accesses

Model	Hidden layers	Neurons per layer	Computing iterations	Training duration (s)
simple ANN-model	12	8	1934	1444
ema-model	11	8	1878	1379
error-class-model	15	27	1551	8655

Table 5. Characteristics of the most succesful ANN-models for random file accesses

Model	Hidden layers	Neurons per layer	Computing iterations	Training duration (s)
simple ANN-model	4	5	1310	222
ema-model	7	5	1106	461
error-class-model	9	22	283	1201

We used the following metrics to quantify the error:

- **MAE**: Mean average error.
- **Avg-MAPE**: The mean absolute percentage error averaged about 12 instances of ANNs with equal parameter values, but different (random) initial network weights.
- **Train-MAPE**: The mean absolute percentage error on the training set.
- **MAPE**: The mean absolute percentage error (on the test set).
- **MSPE**: Mean squared percentage error.
- **RQ3**: Third quartile of prediction errors, which means that three quarters of the test set were predicted with smaller deviation to the true value that this.
- **PMax**: The biggest prediction error in percent.

The prediction errors of our models can be found in the Tables 6 and 7. Listed values refer to the model with the smallest average prediction error. We can observe that for all models values for Avg-MAPE, Train-MAPE and MAPE are close to each other. This means that the training set was representative for the test set with little overfitting, and also the model behavior was stable and did not depend so much on the random initiation of network weights.

Table 6. Results of our models on the sequential file accesses

Model	MAE (s)	Avg-MAPE (%)	Train-MAPE (%)	MAPE (%)	MSPE (%)	RQ3 (%)	PMax (%)
error-class-model	2.0e-05	9	7.7	8.6	14	10	275
ema-model	5.7e-05	14	13.2	13.7	22	18	2336
simple ANN-model	6.0e-05	14	13.6	14.1	22	18	295
Linear regression	7.6e-05	NA	NA	50.8	59	73.7	326

Table 7. Results of our models on the random file accesses

Model	MAE (s)	Avg-MAPE (%)	Train-MAPE (%)	MAPE (%)	MSPE (%)	RQ3 (%)	PMax (%)
error-class-model	0.00103	32	32	31	119	27	4272
simple ANN-model	0.00313	106	104	103	530	70	21786
ema-model	0.00305	421	87	86	619	62	45320
Linear regression	0.00476	NA	NA	5578.4	14185	1158.1	46941

It becomes clear that linear regression is sub-optimal for the prediction of file access times for random file accesses, where its MAPE is about 5578.4 %. For the sequential case the results are better, but still with more than three times higher average percentage error.

The second thing one can take from the results is that the ema-model did not work as intended. Its error values are very close to the simple ANN-model which had less input information to work with. Going a little bit more into detail it is notable that the MAPE is smaller for the ema-models compared to the model without EMA-values. Thus, the additional knowledge was in general useful for better prediction; however, the MSPE and PMax values are higher for the ema-model which means that some predictions were significantly misguided by the EMA-function value. The reason is that the I/O path does not change in a well predictable way (at least from the perspective of a client).

In contrast, the error-class-model worked excellently. The mean average error compared to the model without error classes has been reduced to a third for both use cases. This is a strong confirmation for the thesis that knowledge about I/O paths is crucial for access time prediction. However, it is also clear that the additional parameter of the error class improves the overall performance of the predictor, because it contains knowledge about the measured access times.

To investigate this further, we have to analyze measurements in more detail. To have a further look at the difference that error classes make on the access time prediction, we analyze the predictions of the two models for the case of random reads in Figure 11 and 12.

The model without knowledge about I/O paths is forced to predict some kind of average access time for each set of measurements with equal access parameter values. Therefore, its predictions are in between the two main groups of access time for the higher access sizes. With knowledge about our approximations of I/O paths the error-class-model can discriminate measurements with equal access parameter values and achieve more accurate predictions that way.

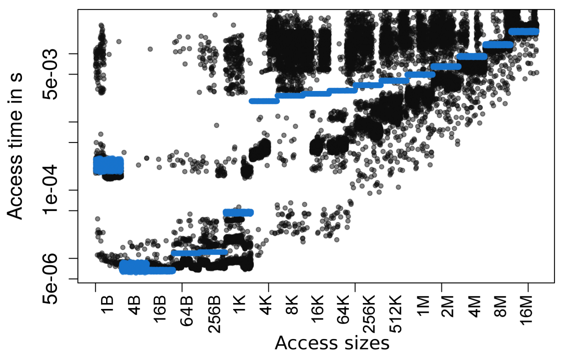

Figure 11. Predicted access times of the simple ANN-model as blue points

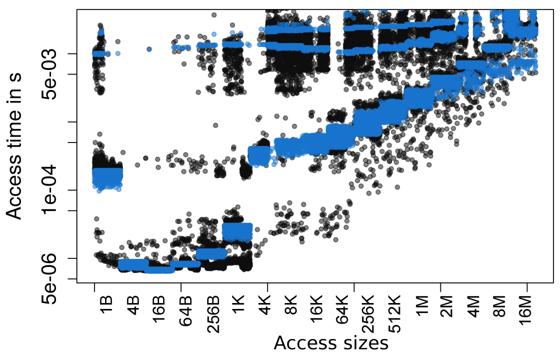

Figure 12. Predicted access times of the error-class-model as blue points

Conclusion and future work

In this paper, we analyzed the performance of a supercomputer's storage system. Using a machine learning approach with artificial neural networks, we developed different models for file access time prediction. Through our study of measured file access times and model results we were able to gain knowledge about the behavior of the storage system. We found out that linear models are not feasible for access time prediction despite the strong correlation of access time to access size. Our models utilizing ANNs achieved much better results than linear regression.

The hypothesis that knowledge about the internal processing of a file access in form of I/O paths is essential for access time prediction was supported by our data. This is because file accesses with equal access parameters can produce strongly deviating access times depending on the I/O path. The model of deriving knowledge about I/O paths by exploiting periodic performance of the storage system was not successful. The additional input information did not lead to a significant improvement of access time predictions.

However, with our method of clustering residues of linear regression for approximations of I/O paths as error classes, we were able to illustrate the importance of I/O paths for access time prediction. The ANN-model with additional input information of error classes was able to reduce its average prediction error to a third compared to the ANN-model with only access parameter values as input information.

For future work a more elaborate approach for exploitation of the periodic storage performance could be used. Crume et al. had success on access time prediction for single hard drives using Fourier analysis [4] or additional sinusoids as input for ANNs [3].

We will research the method to estimate the I/O path based on client measurements further. Residues of other models than linear regression could be used and one could try to assign error classes to I/O paths in a real system – the administrator would have to investigate the error classes and define them according to the storage technology. The method could also be a starting point to develop a tool that provides information on the effectiveness of the used I/O during execution of a program.

References

1. John S. Bucy, Jiri Schindler, Steven W. Schlosser, and Gregory R. Ganger. The DiskSim Simulation Environment Version 4.0 Reference Manual, 2008.
2. Jason Cope, Ning Liu, Sam Lang, Phil Carns, Chris Carothers, and Robert Ross. CODES: Enabling co-design of Multilayer Exascale Storage Architectures. In *Proceedings of the Workshop on Emerging Supercomputing Technologies*, volume 2011, 2011.
3. Adam Crume and Carlos Maltzahn. Latent Frequency Synthesis for Behavioral Hard Disk Drive Access Time Models.
4. Adam Crume, Carlos Maltzahn, Lee Ward, Thomas Kroeger, Matthew Curry, and Ron Oldfield. Fourier-assisted Machine Learning of Hard Disk Drive Access Time Models. In *Proceedings of the 8th Parallel Data Storage Workshop*, PDSW '13, pages 45–51, New York, NY, USA, 2013. ACM. DOI: 10.1145/2538542.2538561.
5. G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989. DOI: 10.1007/BF02551274.
6. Chengjun Dai, Guiquan Liu, Lei Zhang, and Enhong Chen. Storage Device Performance Prediction with Hybrid Regression Models. In *Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, PDCAT'12, pages 556–559, Washington, DC, USA, 2012. IEEE Computer Society. DOI: 10.1109/PDCAT.2012.126.
7. Julian Kunkel, Michaela Zimmer, and Eugen Betke. Using Machine Learning to Predict the Performance of Non-Contiguous I/O, 07 2015. DOI: 10.1109/PDCAT.2012.126
8. Julian M Kunkel. Simulating Parallel Programs on Application and System Level. *Computer Science-Research and Development*, 28(2-3):167–174, 2013.
9. Yonggang Liu, Renato Figueiredo, Dulcardo Clavijo, Yiqi Xu, and Ming Zhao. Towards simulation of parallel file system scheduling algorithms with PFSsim. In *Proceedings of the 7th IEEE International Workshop on Storage Network Architectures and Parallel I/O (May 2011)*, 2011.
10. Raúl Rojas. *Neural Networks: A Systematic Introduction.* Springer-Verlag New York, Inc., New York, NY, USA, 1996.

11. Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27:17–28, 1994.

12. Jan Fabian Schmid. Vorhersage von E/A-Leistung im Hochleistungsrechnen unter der Verwendung von neuronalen Netzen. Bachelor's thesis, Universität Hamburg, 12 2015.

13. Lei Zhang, Guiquan Liu, Xuechen Zhang, Song Jiang, and Enhong Chen. Storage Device Performance Prediction with Selective Bagging Classification and Regression Tree. In *Network and Parallel Computing, IFIP International Conference, NPC 2010, Zhengzhou, China, September 13-15, 2010. Proceedings*, pages 121–133, 2010.