2016 15th IEEE International Conference on Machine Learning and Applications

Filer Response Time Prediction using Adaptively-learned Forecasting Models based on Counter Time Series Data

Saurabh Deshpande
Advanced Technology Group, NetApp
Bangalore, India
Email: dsaurabh@netapp.com

Kumar Dheenadayalan and G Srinivasaraghavan
and VN Muralidhara
Indian Institute of Information Technology - Bangalore, India
Emails: d.kumar@iiitb.org, gsr@iiitb.ac.in, murali@iiitb.ac.in

***Abstract*—Ability to predict the future performance of a storage system is critical for its efficient management. A modern storage system is a very complex combination of hardware and software elements and inferring its *state* from those of its individual components is practically impossible. Moreover, the state of the system undergoes continuous changes due to diverse and evolving workloads, frequent configuration changes and natural degradation in components. This paper presents an approach to predict the future *response time* of a storage system by applying a combination of machine learning techniques on the data of its internal parameters. To the best of our knowledge, this is the first attempt to quantify the response time of a storage system directly from the time-series of its internal parameters. We present a systematic experimental setup for collecting the system data and for measuring its response time while loading the system with different standard workloads. Another novel contribution of the paper is a combination of time-series forecasting model followed by a regression model to predict the response time repeatedly in real-time. The said model is shown to be able to repeatedly predict fairly accurate 15-minutes-ahead response time values using the system data of the past 20-minutes' window. It is shown that the validation calculations typically complete within 30 seconds on an average using a computer of moderate configurations making the presented model applicable for practically reasonable prediction horizons. The design of the model is such that it can be adaptively tuned online at regular intervals as more system data is observed.**

I. Introduction

Monitoring, forecasting and optimizing are the most important tasks for any industrial system. The ability to characterize the state of a system plays a crucial role in these tasks. Methodologies to provide features that help define or quantify system-state are studied in great detail in case of CPU/server monitoring systems. Forecasting the load on small scale servers as well as on large scale Grid/Cloud environments is a well researched area [1]–[3]. There is a lack of similar work on identifying the state of a storage system and forecasting the future trends of key underlying variables that affect the performance of the large-scale environment. A large scale storage system also referred to as a *filer*[1], is a specialized file server having a complex system architecture designed specifically to handle massive parallel data flow [4]. Specialized components and protocols are designed to develop a system of such capabilities. Monitoring each of the physical/logical components, protocols provides large number of performance related information parameters called *counters*. Some examples of counters of physical components are *cpu_busy, nas_throughput*; whereas, *read_latency, net_data_recv* are examples of counters related to logical components. Counter data generated by modern filers is huge as they have numerous unique components each of which outputs tens of counters.

Large scale systems running a variety of applications may exhibit a subset of I/O operations affecting a large subset of counters. Profiling individual applications is of limited help as their characteristics vary with every new version. Hence, identifying the performance of a filer using its counters for a mix of different applications is a big challenge for storage industry. We attempt to solve this by using machine learning algorithms to *predict and forecast the response time of the filer.*

To the best of our knowledge, this is the first study that tries to address the response time prediction using such large scale system data from a filer. There have been many practical scenarios where a filer has become a bottleneck, hampering the performance of a distributed environment. A mechanism to not only learn the current state of a filer from the counters but also forecast their future response will go a long way in calibrating the overall performance of the distributed environment. Insights into the future trend of filer performance also enable taking early actions to achieve optimal filer performance in the long run. The actions to be taken based on such forecasts are not addressed in this paper. Modeling of such a performance forecasting framework is complicated by the diversity in configuration of components, in workloads, in capabilities and capacities of different classes of filers. Availability of large dataset of counters can help in developing a general model for such systems.

Disk level performance modeling and predictions is a well researched area [5]–[8]; but it addresses only a small com-

[1]The term filer and storage system will be used interchangeably in this paper.

978-1-5090-6167-9/16 $31.00 © 2016 IEEE
DOI 10.1109/ICMLA.2016.99

ponent of an enterprise storage system. Indeed, the response time (RT) – as an indicator of performance – is not a well defined term for large scale storage systems as the latter have multi-component architectures. In our experiments, we propose a consistent way of measuring RT and measuring and associating the corresponding counter time series data. This process of counter collection and mapping generated about 4 GB of dataset per day. To the best of our knowledge, no equivalent dataset is available in open literature nor is any attempt reported on using similar data to predict filer RT. Thus, using such large time series datasets is the latest attempt in storage industry to model systems using learning techniques applied to system-generated data.

Our aim is to build a model that is generically applicable to any filer and learns its predictor variables automatically. We also aim the model has the provision of being constantly relearnt to address variations in configurations and workloads. Dimensionality and size of the dataset are challenges for this approach.

Rest of the paper is organized as follows. Section II gives an insight on the past literature. A detailed description of our experimental setup and the dataset is provided in Section III followed by the proposed solution in Section IV. We present the experimental results for validation of the model in Section V and conclude in Section VI.

II. Related Work

Past research related to modeling and performance improvement of storage systems has mainly focused at disk level [5], [6], [9]–[12]. Majority of this work models the performance of the disk drives based on the workload characteristics [8], [12]. Using only workload characteristics to model a storage system is not practically feasible as the diversity of workload characteristics that can be generated by a large number of scientific, I/O intensive and Big Data applications is increasing.

White Box approaches for storage systems where system parameters are used for modeling and prediction are also focused at a small scale, viz., disk drive. Disk parameters are used in failure predictions; but are not very popular for performance modeling [7], [13]. Modeling using a combination of system and workload parameters to predict system performance [5], [12] allows for generalizing the model to evaluate storage system for a large set of different configurations. Machine learning algorithms have also been used in modeling different classes of storage devices, like SSD, HDD [6], [14], for capturing their response for a specified application. Learning algorithms have played a key role in many of the performance modeling techniques used in the literature [5], [6], [8]. ARIMA time series models have been used to predict temporal patterns of I/O requests [11].

Attempts in the past for solving performance modeling problems of distributed storage [15], [16] address the modeling problem in a traditional way by considering RT as a function of throughput and the number of requests from client. None of the past literature in this domain [6], [15], [16] considers the complex aspects of a storage system. The set of features

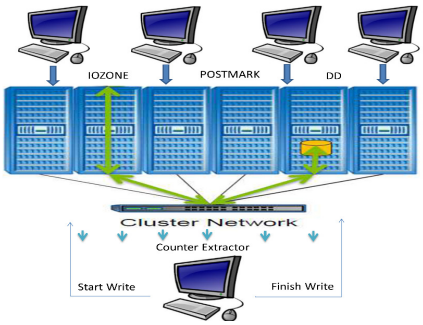


Fig. 1: Experimental setup for data collection

in these works are restricted to read/write operations, latency, throughput and number of requests. Such models are not scalable for modern filers having numerous components. Two of the earlier works [17], [18] used the counter data for analysis of filer performance. However, application of learning algorithms on counter data in [18] was aimed at analyzing and identifying workload on the filer. In [17], counter dataset was used for classification of *response-state* of a filer. The latter was based on a simplified conception of only two response-states of a filer, viz., *normal* and *saturated*. In contrast, the proposed framework in the present work is designed for outputting actual RT values of filers. Thus, the approach proposed here gives practically more useful results than the one in [17].

III. Experimental Setup and Data Description

Our scheme of systematic data collection from an industrial filer is pictorially represented in Figure 1.

The set of clients at the top continuously generated I/O load on filer by running benchmarking/real world applications. We used 5 such clients for workload generation. I/O loads consisted of maximum 2 instances of *IOZONE* tool [19] running in parallel, each launched at random time intervals. Each instance performed multiple iterations of 13 different classes of I/O operations [20] and each iteration chose different I/O block sizes ranging from 4KB to 16MB. The intention was to simulate the behavior of different classes of applications, whereas multiple instances were aimed at simulating multi-user behavior. A single *dd* utility [21] was used for writing 1GB of data of varying block-sizes (ranging from 1KB to 1GB) in each of its iterations. *Postmark* tool [22] was used for generating meta-data operations on millions of files with a maximum of 2 instances running in parallel. Table I describes the overall usage of each tool in our experiments. Maximum amount of data written was capped at 100GB per instance of each tool to prevent exhaustion of disk space on the filer.

While the filer was running different types of workloads, a counter extraction module was run in parallel on one of the clients as shown at the bottom of Figure 1. The counter extractor module was programmed to write 1 GB of data to the filer (*write cycle*), measure the completion time for the write operation and query values of counters from the filer at

Tools	IOZONE	Postmark	dd
No. of instances	2	2	1
I/O operation(s)	13 types	create-write-read-delete	write-delete
Duration	57 hours	20 hours	57 hours
Block size(s)	4KB – 16MB	NA	100KB – 1GB
File size(s)	1GB – 20GB	1KB – 50KB	1 GB

TABLE I: Summary of workload generating operations

regular time-intervals s while the operation completed. Write operation was chosen as a benchmark operation for measuring the filer RT since it is one of the costliest I/O operations.

Thus, the setup was designed to generate pairs of filer RT and counter TS data over corresponding duration for multiple expensive filer operations of interest (viz., writes), when the filer was already loaded with multiple standard/real-life workloads. Thousands of such write cycles were carried out to collect a sufficiently large dataset for our analysis and also to allow different types of workloads to push the filer response to its maximum limit multiple times. No equivalent dataset is available in open literature.

A key feature of our approach is that we are not measuring the response time in a traditional sense as seen on disks or CPUs. Given the complex architecture of filers, it is not easy to correlate latencies or RTs of individual components to those of filer as a whole. What is more relevant for a user of the filer is its response as a whole. Another salient feature of our approach is that the RT is measured at the client level. This again ensures that the dependent variable of the model, viz., the RT, gives a close indication of the actual experience – including typical network latencies – of the users/servers/applications that run workloads on the filer.

The dataset is used to train a machine learning model to predict future values of the RT as described in the next section. The training can be done offline using previously collected data or can be repeated online after multiple runs of validation cycles. For validation of the model, the same procedure as described above was repeated to load the filer and to measure the counter and RT values.

Table II lists the key details of the dataset used to build the learning model described in the next section.

IV. Proposed Model for Predicting Response Time

In this section, we describe our proposed model for predicting filer RT based on values of its counters, which is learnt using the dataset described in Section III.

As described in Section I, the main aim of this work is to utilize counter data as representative of the *state* of the filer using which to quantify filer RT. Given the very large no. of counters recorded for a filer, it is quite reasonable to expect only a small subset of these to be relevant to filer RT. A straightforward way to select a set of *relevant* counters would be to check the correlation between the TS of RT and of each counter recorded during the training [23]. Pearson's

$\mathcal{C}$	set of counters measured; $\lvert\mathcal{C}\rvert = 48,663$
s	sampling interval for measurement of each counter TS in both training and testing phases
n_w	no. of write operations in training phase
$\mathbf{r}$	vector of RT of all write operations in training phase, i.e., $\lvert\mathbf{r}\rvert = n_w$
w_j	time-interval of the j^{th} write operation in training phase
$\mathbf{C}_i^{w_j}$	TS vector of i^{th} counter collected during w_j
W	time-window for which measured counter TS values are input to the learned model in testing phase
R	time interval after which a testing run is repeated online
n_v	total no. of test runs
H	prediction horizon chosen for the learning model in testing phase

TABLE II: Parameters of the model

correlation coefficient is one of the most widely used measures to calculate similarity between two TS [24]:

$$\rho(\mathcal{X},\mathcal{Y}) = \text{Cov}(\mathcal{X},\mathcal{Y}) \Big/ \Big(\sqrt{\text{Var}(\mathcal{X})}\sqrt{\text{Var}(\mathcal{Y})}\Big).$$

We noted that majority of the counters have a very small correlation (in magnitude) with the RT. In fact, the no. of counters having a correlation of greater than 0.2 with the RT is only 922, i.e., 2% of the total no. of counters, corroborating the aforementioned intuition. A lower correlation threshold will select relatively more counters as *highly relevant*, leading to higher time-complexity of any algorithm making use of them. Hence, any choice of a correlation threshold has to take into account the trade-off between prediction accuracy and the time-complexity of the forecast. We chose a correlation threshold of 0.45, which is the 99^{th} percentile of the distribution above and which resulted in a subset of 68 counters chosen for model development. More systematic ways for (automatically) selecting a correlation threshold [23] to attain prescribed levels of model performance and/or complexity will be investigated in future work.

The aim of our machine learning approach is two-fold, viz., predict filer RT based on *state* of the filer *as represented by its counters* and predict the future RT using historical and present values of the said data. We propose to build such a model using a combination of a forecaster for future values of individual counter TS and a regressor on counter values to predict the corresponding RT. Thus, the training phase consists of learning the regression model using the (counter TS, RT) dataset collected as described in Section III. In the testing phase, first the counter TS data is collected over a time-window W and using only these values, counter values H-minutes ahead are forecast, for example, using an approach similar to [25]. The said forecast values are then input to the regression model learned earlier to obtain predicted RT H-minutes ahead.

We chose the widely used ARIMA [26] model for the counter TS forecaster and Random Forest Regression (RFR)

[27] model for predicting the RT from the counter values. The approach is summarized below more formally.

- Extracting the set of counters highly correlated with RT:

$$\mathcal{C}_r := \{C_i \mid C_i \in \mathcal{C},\ \rho(\mathbf{r}, \mathbf{C}_i^{w_j}) \geq t_c\}, \tag{1}$$

where t_c is a threshold to be chosen for determining high-correlation. Let $\mathcal{I}_{\mathcal{C}_r}$ denote the index set corresponding to $\mathcal{C}_r$.

- Offline training of RFR model for predicting RT from instantaneous counter values:

$$\text{RFRM} \leftarrow \text{learnRFRegrssion}(\mathbf{r}[j]; \mathbf{C}_i^{w_j}[n_w]; \alpha, \beta), \tag{2}$$

$\forall i \in \mathcal{I}_{\mathcal{C}_r}, \forall j \in \mathcal{I}_{n_w}$, where $\mathbf{r}$ and $\mathbf{C}_i^{w_j}$ are as defined earlier in Table II, α is the number of trees to be built as part of the forest and β is the number of counters considered in each split during the tree construction. Note that the counter TS values $\mathbf{C}_i^{w_j}$ are averaged over w_j as a preprocessing step for RFRM. Thus, each w_j interval contributes one tuple of {counters, RT} to training data of RFRM.

- Online forecasting of individual counter values H-minutes ahead in the testing phase:

$$\bar{C}_i(t+H) = \text{ARIMA}(\mathbf{C}_i^{w_t}; p, d, q), \tag{3}$$

where

$$w_t := (t-W, t]\ t,\ i \in \mathcal{I}_{\mathcal{C}_r}\},$$

and p, d, q are the standard ARIMA coefficients [28].

- Online prediction of RT using the forecast counter values:

$$\bar{r}(t+H) = \text{RFRM}(\bar{C}_1(t+H), \ldots, \bar{C}_i(t+H), \ldots), \tag{4}$$

$\forall i \in \mathcal{I}_{\mathcal{C}_r}$.

Note that the first two steps above need not be only offline. Indeed, by repeating these steps periodically with a period sufficiently longer than that of a single-run of training and testing, the entire learning is repeatedly adapted to variations seen in TS of counters and filer RT. Thus, this approach can be implemented as an online self-tuning model.

Standard error measures like RMSE and MAE are used for evaluating RT predictions whereas error measures relating to time series forecasting like SMAPE [29] are used to evaluate strength of counter TS forecasts. For the case at hand,

- Mean Absolute Error:

$$\text{MAE}_{RT} = \frac{1}{n_v} \sum_{t \in \mathcal{T}} |\bar{r}(t+H) - r(t+H)|\ ,$$

- Root Mean Square Error:

$$\text{RMSE}_{RT} = \sqrt{\frac{1}{n_v} \sum_{t \in \mathcal{T}} (\bar{r}(t+H) - r(t+H))^2}\ ,$$

- Symmetric Mean Absolute Percentage Error :

$$\text{SMAPE}_{C_i} = \frac{1}{n_v} \sum_{t \in \mathcal{T}} \frac{|\bar{C}_i(t+H) - C_i(t+H)|}{\frac{\bar{C}_i(t+H) + C_i(t+H)}{2}}\ ,$$

where $i \in \mathcal{I}_{\mathcal{C}_r}$,

$\lvert\mathcal{C}\rvert$	$48,663$	s	10 sec	n_v	$4,608$
n_w	$3,244$	t_c	0.45	α	200
$\lvert\mathcal{C}_r\rvert$	68	H	15 minutes	β	8
W	20 minutes	R	1 minute		

TABLE III: Nominal values of parameters used to test the model

and where

$$\mathcal{T} := \{R, 2R, \ldots, n_v * R\}.$$

It is clear that the model proposed above attains all the goals mentioned in Section I. In particular, the model is generic enough to be applicable to any filer and the choice of its predictor variables, viz., counters, is automated. The accuracy and complexity of the model can be tuned using simple parameters such as sampling period s, correlation threshold t_c, prediction horizon H and length of data-window for forecasting W.

We present experimental results of the performance of the model in the next section.

V. Experiments with the model

Performance of the model was tested under multiple scenarios by varying different parameters of the model as well as the setup. Data extraction, preprocessing, regression modeling and forecasting were tested on a live filer.

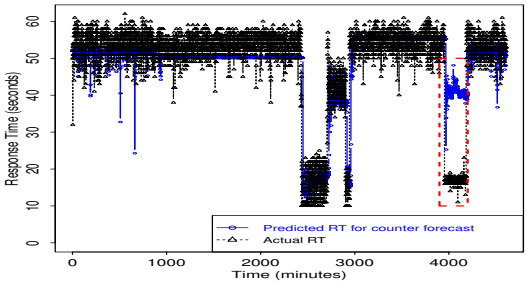
We first present the results of the experiment for the values of parameters listed in Table III.

The training phase, in which counter TS were collected and preprocessed for RFRM construction as explained in Section IV, lasted for over 57 hours. Average time for forecasting of 68 counters over all iterations was 31 seconds, which is less than inter-iteration window (R) of 1 minute. Thus, the counter forecasts for RT predictions were available within each test interval. The testing phase lasted for over 77 hours.

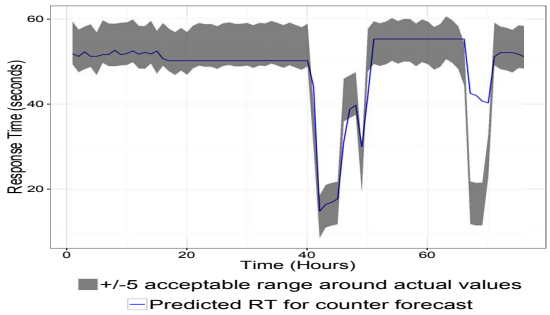
Maximum value of forecasting error SMAPE was 46.9%, while median and mean values were 18.69% and 16.68%, respectively. Recalling that SMAPE has a range of $(0\%, 200\%)$ [29], forecasting errors for the majority of counters are clearly on the lower side of the spectrum.

We next analyze the impact of SMAPE on the overall quality of RT prediction. Figures 2a and 2b show the actual and predicted values of the RT (in seconds) recorded for every write operation over the entire testing phase. To enhance visualization and analysis of different regions on the plot, we average the actual and prediction values of RT over a period of 1 hour as shown in Figure 2b. The grey region in the plot, shows a close *acceptable range* of ± 5 units around the actual RT values. The plot clearly shows that for majority of the testing phase, viz., 73 hours out of total 77 hours, the predicted RT values lay in the acceptable range. In more quantitative terms, the MAE and RMSE error values for RT predictions over the 77 hour testing period were 4.847 and 7.949, respectively, which are reasonably low numbers.

Certain regions with high deviations from actual RT values can be seen in the latter half of the test phase marked with



(a) Predicted (15-minutes ahead) vs actual values of RT



(b) Predicted (15-minutes ahead) vs actual values of RT averaged over successive 1-hour time-windows

Fig. 2: Comparison of predicted and actual RT values

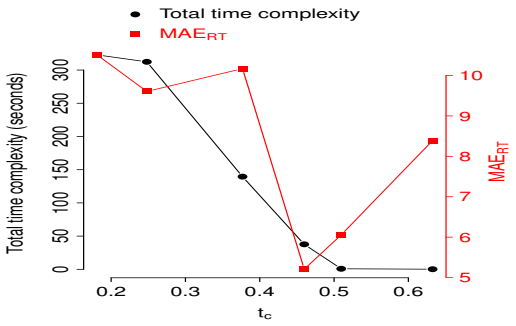


Fig. 3: Effect of variation in t_c on performance of the model

a rectangular region in Figure 2a. Due space restrictions, it is not possible to present a detailed analysis of these deviations here. Next, we present results on effect of variation in key parameters of the model on its performance.

A. *Effect of variation in t_c, W and H on model performance*

Changing the value of correlation threshold t_c causes the no. of counters participating in the model to change, thereby directly affecting the counter forecast and RT predictions.

The impact of change in t_c on the (average) total evaluation time, i.e. ARIMA forecasting of $|\mathcal{C}_r|$ counters and RFRM prediction, and on the prediction error MAE_{RT} is shown in Figure 3 (Y-axis$_{\text{left}}$ and Y-axis$_{\text{right}}$, respectively). Nearly 100% increase in time-complexity was observed for $t_c = 0.17$ (80^{th} percentile) compared with $t_c = 0.65$ (99.9^{th} percentile). Since a model with total evaluation time per iteration greater than the chosen value of R is not desirable, the *feasible* percentile values of t_c for our setup – with value of R chosen to be 1 minute – are above 0.42 (97^{th} percentile value). As t_c is reduced from 0.65 (99.9^{th} percentile value) till 0.46 (99^{th} percentile), a gradual decrease was initially observed in MAE_{RT} over a short set of 200 test intervals. However, MAE_{RT} started increasing after further decrease in t_c and reached as high as 10.2% for 0.17 (80^{th} percentile value), as seen in Figure 3. Thus, it seems that using more number of counters does not contribute to discriminatory power of the predictive model beyond a limit. The 99^{th} percentile value for correlation coefficient provided the best practical trade-off between total time-complexity versus accuracy of RT predictions.

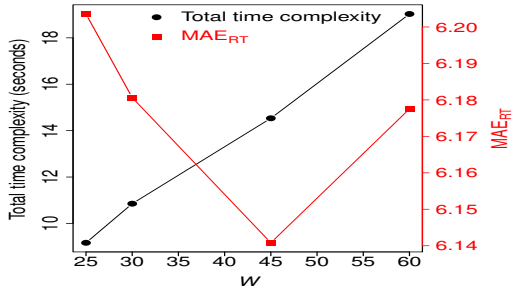


Fig. 4: Effect of change in W on the testing time and MAE_{RT}

For a fixed t_c, H and R of 0.45, 15 minutes and 1 minute respectively, the effect of changing data-window used for counter forecasting (W) on MAE_{RT} is shown in Figure 4. Y-axis$_{\text{left}}$ represents the total time taken to forecast counters and predict RT while Y-axis$_{\text{right}}$ represents the MAE_{RT}. The errors were measured over $2,500$ test intervals for each value of W. Marginal variation was observed in the error rates (from 6.14 to 6.20) whereas high variation was observed in the *total time* of RT predictions (from 6 hours 21 minutes to 13 hours 12 minutes). However, for all four values of W, the average time taken for each RT prediction varied between 9 to 19 seconds, which is within the chosen value for $R = 1$ minute.

A longer forecasting horizon H will intuitively result in lower forecasting accuracy, which in turn, will reduce the accuracy of the RT predictions. We tested the effect of 5 different values of H between 20 to 60 minutes on the model performance for fixed values of t_c, W and R of 0.45, 60 minutes and 1 minute, respectively. In Figure 5, Y-axis$_{\text{left}}$ represents the total time taken to forecast counters and predict RT while Y-axis$_{\text{right}}$ represents the MAE_{RT} with corresponding values of H on X-axis. The values were measured over $2,500$ test intervals for each value of H. As expected, we observe that the prediction error MAE_{RT} increases with H, the variation being especially prominent beyond a forecasting horizon of 30 minutes. The variation in time complexity on the other hand is quite marginal (between 19.1 sec to 20.1 sec per instance).

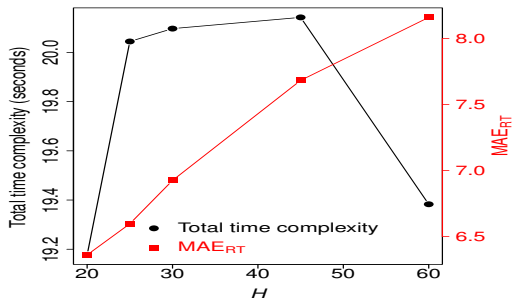


Fig. 5: Testing time and MAE_{RT} for different values of forecasting horizon H

VI. Conclusion

This work reports a first attempt to predict future response time of industrial filers. We present a novel combination of regression and forecasting algorithms to efficiently and accurately forecast the response time of a filer. Using the vast counter TS data available from filer, our proposed model learnt the mapping between counter TS and the response times and was able to predict the latter in advance. Experiments carried out extensively over multiple days were evaluated for the effect of variation in different factors influencing the RT predictions.

The future directions of research would focus on the choice of prediction and forecasting algorithms as well as more complex features of the data. Dynamic correlation threshold selection would be developed for automating the framework. At present, all tasks for profiling filer performance are manual and time-consuming in nature. Using the RT predictions in optimizing the filer performance will have a major impact on not just the filer but also on other entities on the distributed computing environment of which the filer is a part. Proposed system holds the potential for numerous useful applications, e.g., to preempt a potential high-load scenario, to identify the right time-slot for data migration and so on.

References

[1] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 1–11.

[2] Y. Zhang, W. Sun, and Y. Inoguchi, "CPU load predictions on the computational grid *," in 6^{th} *IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, vol. 1, 2006, pp. 321–326.

[3] ——, "Predicting running time of grid tasks based on CPU load predictions," in *Proceedings of the* 7^{th} *IEEE/ACM International Conference on Grid Computing (GRID)*, 2006, pp. 286–292.

[4] D. van der Ster and A. Wiebalck, "Building an organic block storage service at CERN with Ceph," *J. Phys.: Conf. Ser.*, vol. 513, pp. 42–47, 2014. [Online]. Available: https://cds.cern.ch/record/2026340

[5] A. Aldahlawi, E. El-Araby, S. Suboh, and T. El-Ghazawi, "Modelling the performance of an ssd-aware storage system using least squares regression," in *Proceedings of the* 9^{th} *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, 2011, pp. 181–187.

[6] C. Dai, G. Liu, L. Zhang, and E. Chen, "Storage device performance prediction with hybrid regression models," in 13^{th} *International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2012, pp. 556–559.

[7] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population," in *Proceedings of the* 5^{th} *USENIX Conference on File and Storage Technologies (FAST)*, 2007, pp. 2–2.

[8] M. Wang, K. Au, A. Ailamaki, A. Brockwell, C. Faloutsos, and G. R. Ganger, "Storage device performance prediction with CART models," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 412–413, 2004.

[9] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *Proceedings of the* 18^{th} *International Conference on Machine Learning (ICML)*, 2001, pp. 202–209.

[10] Y. Lu, Y. Chen, P. Amritkar, R. Thakur, and Y. Zhuang, *A New Data Sieving Approach for High Performance I/O*. Springer Netherlands, 2012, pp. 111–121.

[11] N. Tran and D. A. Reed, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 4, pp. 362–377, 2004.

[12] L. Zhang, G. Liu, and E. Chen, "A hybrid approach based on regression tree and radial-based function network for dynamic storage device performance prediction," in *International Conference on Computer Science and Service System (CSSS)*, 2011, pp. 21–25.

[13] Z. Lan, J. Gu, Z. Zheng, R. Thakur, and S. Coghlan, "A study of dynamic meta-learning for failure prediction in large-scale systems," *J. Parallel Distrib. Comput.*, vol. 70, no. 6, pp. 630–643, 2010.

[14] M. P. Mesnier, M. Wachs, R. R. Sambasivan, A. X. Zheng, and G. R. Ganger, "Modeling the relative fitness of storage," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2007, pp. 37–48.

[15] K. T. Pollack and S. M. Uttamchandani, "Genesis: A scalable self-evolving performance management framework for storage systems," in 26^{th} *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006, pp. 33–33.

[16] E. Thereska, M. Abd-El-Malek, J. J. Wylie, D. Narayanan, and G. R. Ganger, "Informed data distribution selection in a self-predicting storage system," in *Proceedings of the* 3^{rd} *International Conference on Autonomic Computing, (ICAC)*, 2006, pp. 187–198.

[17] K. Dheenadayalan, V. N. Muralidhara, P. Datla, G. Srinivasaraghavan, and M. Shah, "Premonition of storage response class using skyline ranked ensemble method," in 21^{st} *International Conference on High Performance Computing (HiPC)*, 2014, pp. 1–10.

[18] S. R. Sandeep, M. Swapna, T. Niranjan, S. Susarla, and S. Nandi, "Cluebox: A performance log analyzer for automated troubleshooting," in *Proceedings of the* 1^{st} *USENIX Conference on Analysis of System Logs (WASL)*, 2008.

[19] D. Capps and T. McNeal. Iozone. http://www.iozone.org/. [Online; accessed 15-Feb-2016].

[20] ——. Analyzing nfs client performance with iozone. http://www.iozone.org/docs/NFSClientPerf_revised.pdf. [Online; accessed 15-Feb-2016].

[21] Roman. How to use 'dd' to benchmark your disk or CPU? https://romanrm.net/dd-benchmark. [Online; accessed 12-Dec-2015].

[22] J. Katcher, "Postmark: A new file system benchmark," 1997, technical Report TR3022. Network Applicance Inc. 1997.

[23] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the* 17^{th} *International Conference on Machine Learning (ICML)*, 2000, pp. 359–366.

[24] R. S. Tsay, *Analysis of Financial Time Series*. Wiley-Interscience, 2005.

[25] R. J. Hyndman and Y. Khandakar, "Automatic Time Series Forecasting: The forecast Package for R," *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008.

[26] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*, ser. Springer Texts in Statistics. Springer, 2010.

[27] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[28] G. E. P. Box and G. M. the, *Time Series Analysis: Forecasting and Control*. Prentice Hall, USA, 1994.

[29] J. Scott Armstrong, *Long-range Forecasting: From Crystal Ball to Computer*. John Wiley, 1985.