12th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). Volendam, The Netherlands. October 5-7, 2004.

Storage Device Performance Prediction with CART Models

Mengzhi Wang, Kinman Au, Anastassia Ailamaki,
Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

Storage device performance prediction is a key element of self-managed storage systems. This work explores the application of a machine learning tool, CART models, to storage device modeling. Our approach predicts a device's performance as a function of input workloads, requiring no knowledge of the device internals. We propose two uses of CART models: one that predicts per-request response times (and then derives aggregate values) and one that predicts aggregate values directly from workload characteristics. After being trained on the device in question, both provide accurate black-box models across a range of test traces from real environments. Experiments show that these models predict the average and 90th percentile response time with a relative error as low as 19%, when the training workloads are similar to the testing workloads, and interpolate well across different workloads.

1. Introduction

The costs and complexity of system administration in storage systems [9, 19, 6] make automation of administration tasks a critical research challenge. One important aspect of administering self-managed storage systems, particularly large storage infrastructures, is deciding which data sets to store on which devices. Automatic storage provision tools, such as Ergastulum [1], rely on efficient and accurate device models in making such decisions. To find an optimal or near optimal solution requires the ability to predict how well each device will serve each workload, so that loads can be balanced and particularly good matches can be exploited.

Researchers have long utilized performance models for such prediction to compare alternative storage device designs. Given sufficient effort and expertise, accurate simulations (e.g., [4, 13]) or analytic models (e.g., [10, 14, 15]) can be generated to explore design questions for a particular device. Unfortunately, in practice, such time and expertise is not available for deployed infrastructures, which are often comprised of numerous and distinct device types, and their administrators have neither the time nor the expertise needed to configure device models.

This paper attacks this obstacle by providing a black-box model generation algorithm. By "black box," we mean that the model (and model generation system) has no information about the internal components or algorithms of the storage device. Given access to a device for some "training period," the model generation system learns a device's behavior as a function of input workloads. The resulting device model approximates this function using existing machine learning tools. Our approach employs the Classification And Regression Trees (CART) tool because of its efficiency and accuracy. CART models, in a nutshell, approximate functions on a multi-dimensional Cartesian space using piece-wise constant functions.

Such learning-based black box modeling is difficult for two reasons. First, all the machine learning tools we have examined use vectors of scalars as input. Existing workload characterization models, however, involve parameters of empirical distributions. Compressing these distributions into a set of scalars is not straightforward. Second, the quality of the generated models depends highly on the quality of the training workloads. The training workloads should be diverse enough to provide high coverage of the input space.

This work develops two ways of encoding workloads as vectors: a vector per request or a vector per workload. The two encoding schemes lead to two types of device models, operating at the per-request and per-workload granularities, respectively. The request-level device models predict each request's response time based on its per-request vector, or "request description." The workload-level device models, on the other hand, predict aggregate performance directly from per-workload vectors, or "workload descriptions." Our experiments on a variety of real world workloads have shown that these descriptions are reasonably good at capturing workload performance from both single disks and disk arrays. The two CART-based models have a median relative error of 17% and 38%, respectively, for average response time prediction, and 18% and 43% respectively for the 90th percentile, when the training and test-

ing traces come from the same workload. The CART-based models also interpolate well across workloads.

The remainder of this paper is organized as follows. Section 2 discusses previous work in the area of storage device modeling and workload characterization. Section 3 describes CART and its properties. Section 4 describes two CART-based device models. Section 5 evaluates the models using several real-world workload traces. Section 6 concludes the paper.

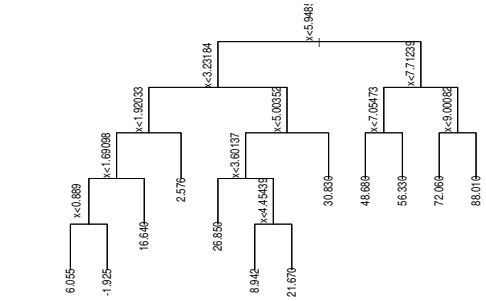
2. Related Work

Performance modeling has a long and successful history. Almost always, however, thorough knowledge of the system being modeled is assumed. Disk simulators, such as Pantheon [18] and DiskSim [4], use software to simulate storage device behavior and produce accurate per-request response times. Developing such simulators is challenging, especially when disk parameters are not publicly available. Predicting performance using simulators is also resource intensive. Analytical models [5, 10, 11, 14, 15] are more computationally efficient because these models describe device behavior with a set of formulae. Finding the formula set requires deep understanding of the interaction between storage devices and workloads. In addition, both disk simulators and analytical models are tightly coupled with the modeled device. Therefore, new device technologies may invalidate existing models and require a new round of model building.

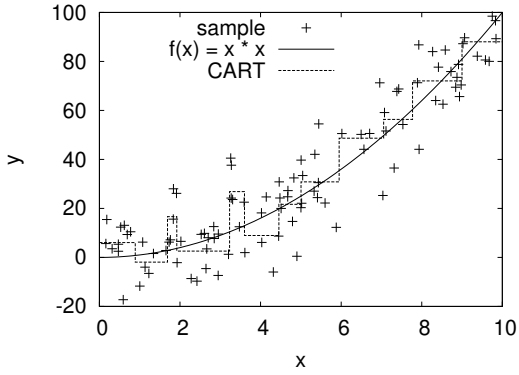
Our approach uses CART, which treats storage devices as black boxes. As a result, the model construction algorithm is fully automated and should be general enough to handle any type of storage device. The degenerate forms of "black-box models" are performance specifications, such as the maximum throughput of the devices, published by device manufacturers. The actual performance, however, will be nowhere near these numbers under some workloads. Anderson's "table-based" approach [2] includes workload characteristics in the model input. The table-based models remember device behavior for a range of workload and device pairs and interploates among tables entries in predicting. Our approach improves on the table-based models by employing machine learning tools to capture device behavior. Because of the good scalability of the tools to high dimensional datasets, we are able to use more sophisticated workload characteristics as the model input. As a result, the models are more efficient in both computation and storage.

3. Background: CART Models

This section gives a brief introduction of the CART models. A detailed discussion of CART is available in [3].



(a) Fitted tree



(b) Data points and regression line

Figure 1. A sample CART model.

3.1. CART Models

CART modeling is a machine learning tool that can approximate real functions in multi-dimensional Cartesian space. (It can also be thought of as a type of non-linear regression.) Given a function $Y = f(X) + \epsilon$, where $X \in \Re^d$, $Y \in \Re$, and ϵ is zero-mean noise, a CART model approximates Y using a piece-wise constant function, $\hat{Y} = \hat{f}(X)$. We refer to the components of X as features in the following text. The term, ϵ, captures the intrinsic randomness of the data and the variability contributed by the unobservable variables. The variance of the noise could be dependent on X. For example, the variance of response time often depends on the arrival rate.

The piece-wise constant function $\hat{f}(X)$ can be visualized as a binary tree. Figure 1(a) shows a CART model constructed on the sample one-dimensional data set in (b). The data set is generated using

$$y_i = x_i^2 + \epsilon_i, \qquad i = 1, 2, \ldots, 100,$$

where x_i is uniformly distributed within (0,10), and ϵ_i follows a Guassian distribution of $N(0, 10)$. The leaf nodes correspond to disjoint hyper-rectangles in the feature vector space. The hyper-rectangles are degenerated into intervals for one-dimensional data sets. Each leaf is associated

with a value, $\hat{f}(X)$, which is the prediction for all Xs within the corresponding hyper-rectangle. The internal nodes contain split points, and a path from the root to a leaf defines the hyper-rectangle of the leaf node. The tree, therefore, represents a piece-wise constant function on the feature vector space.

3.2. CART Model Properties

CART models are computationally efficient in both construction and prediction. The construction algorithm starts with a tree with a single root node corresponding to the entire input vector space and grows the tree by greedily selecting the split point that yields the maximum reduction in mean squared error. Each prediction involves a tree traversal and, therefore, is fast.

CART offers good interpretability and allows us to evaluate the importance of various workload characteristics in predicting workload performance. A CART model is a binary tree, making it easy to plot on paper as in Figure 1(a). In addition, one can evaluate a feature's importance by its contribution in error reduction. In a CART model, we use the sum of the error reduction related to all the appearances of a feature as its importance.

4. Predicting Performance with CART

This section presents two ways of constructing device models based on CART models.

4.1. Overview

Our goal is to build a model for a given storage device which predicts device performance as a function of I/O workload. The device model receives a workload as input and predicts its aggregate performance. We define a workload as a sequence of disk requests, with each request, r_i, uniquely described by four attributes: arrival time ($ArrivalTime_i$), logical block number (LBN_i), request size in number of disk blocks ($Size_i$), and read/write type (RW_i). The storage device could be a single disk, a disk array, or some other like-interfaced component. The aggregate performance can be either the average or the 90-th percentile response time.

Our approach uses CART to approximate the function. We assume that the model construction algorithm can feed any workload into the device to observe its behavior for a certain period of time, also known as "training." The algorithm then builds the device model based on the observed response times. Model construction does not require any information about the internals of the modeled device. Therefore, the methodology is general enough to model any device, even if the modeling approach may not be.

Regression tools are a natural choice to model device behavior. Such tools are designed to model functions on multi-dimensional space given a set of samples with known output. The difficulty is to transform workloads into data points in a multi-dimensional feature space. We explore two ways to achieve the transformation as illustrated in Figure 2. A request-level model represents a request r_i as a vector R_i, also known as the "request description," and uses CART models to predict per-request response times. The aggregate performance is then calculated by aggregating the response times. A workload-level model, on the other hand, represents the entire workload as a single vector W, or the "workload description," and predicts the aggregate performance directly from W. In both approaches, the quality of the input vectors is critical to the model accuracy. The next two sections present the request and workload descriptions in detail.

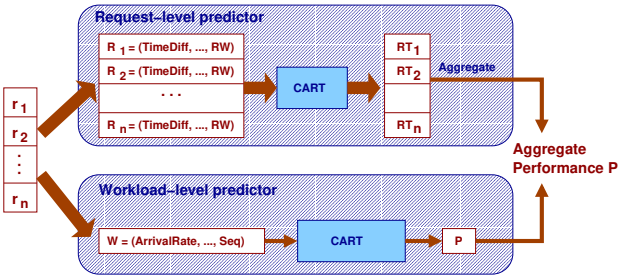


Figure 2. Two types of device models.

4.2. Request-Level Device Models

This section describes the CART-based request-level device model. This model uses a CART model to predict the response times of individual requests based on request descriptions. The model, therefore, is able to generate the entire response time distribution and output any aggregate performance measures.

Our request description R_i for request r_i contains the following variables:

$$\begin{aligned} R_i = \{ \; & TimeDiff_i(1), \; \ldots, \; TimeDiff_i(k), \\ & LBN_i, \; LBNDiff_i(1), \; \ldots, \; LBNDiff_i(l), \\ & Size_i, \; RW_i, \\ & Seq(i) \; \}, \end{aligned}$$

where $TimeDiff_i(k) = ArrivalTime_i - ArrivalTime_{i-2^{k-1}}$ and $LBNDiff_i(l) = LBN_i - LBN_{i-l}$. The first three groups of features capture three components of the response time, and $Seq(i)$ indicates whether the request is a sequential access. The first $(k+1)$ features measure the temporal burstiness of the workload when r_i arrives, and support prediction of the queuing

time. We allow the $TimeDiff$ features to exponentially grow the distance from the current request to history request to accommodate large bursts. The next $(l+1)$ features measure the spatial locality, supporting prediction of the seek time of the request. $Size_i$ and RW_i support prediction of the data transfer time.

The two parameters, k and l, determine how far we look back for request bursts and locality. Small values do not adequately capture these characteristics, leading to inferior device models. Large values, on the other hand, leads to a higher dimensionality, meaning the need for a larger training set and a longer training time. The optimal values for these parameters are highly device specific, and Section 5.1 shows how we select the parameter values in our experiments.

4.3. Workload-Level Device Models

The workload-level model represents the entire workload as a single workload description and predicts aggregate device performance directly. The workload description W contains the following features.

$$W = \{ \begin{array}{l} \text{Average arrival rate,} \\ \text{Read ratio,} \\ \text{Average request size,} \\ \text{Percentage of sequential requests,} \\ \text{Temporal and spatial burstiness,} \\ \text{Correlations between pairs of attributes} \end{array} \}.$$

The workload description uses the entropy plot [16] to quantify temporal and spatial burstiness and correlations between attributes. Entropy values are plotted on one or two attributes against the entropy calculation granularity. The increment of the entropy values characterizes how the burstiness and correlations change from one granularity to the next. Because of the self-similarity of I/O workloads [7], the increment is usually constant, allowing us to use the entropy plot slope to characterize the burstiness and correlations. Please refer to [17] for how we apply entropy plot to quantify the burstiness and correlations.

The workload-level device model offers fast predictions. The model compresses a workload into a workload description and feeds the description into a CART model to produce the desired performance measure. Feature extraction is also fast. To predict both the average and 90th percentile response time, the model must have two separate trees, one for each performance metric.

Workload modeling introduces a parameter called "window size." The window size is the unit of performance prediction and, thus, the workload length for workload description generation. For example, we can divide a long trace into one-minute fragments and use the workload-level model to predict the average response time over one-minute intervals. Fragmenting workloads has several advantages. First, performance problems are usually transient. A "problem" appears when a large burst of requests arrive and disappears quickly after all the requests in the burst are served. Using the workload in its entirety, on the other hand, fails to identify such transient problems. Second, fragmenting the training trace produces more samples for training and reduces the required training time. Windows that are too small, however, contain too few requests for the entropy plot to be effective. We use one-minute windows in all of our experiments.

4.4. Comparison of Two Types of Models

There is a clear tradeoff between the request-level and workload-level device models. The former is fast in training and slow in prediction, and the latter is the opposite.

An item for future research is the exploration of the possibility of combining the two models to deliver ones that are efficient in both training and prediction.

5. Experimental Results

This section evaluates the CART-based device models presented in the previous section using a range of workload traces.

Devices. We model two simulated devices: a single disk and a disk array. The single disk is a 9GB Atlas 10K disk with an average rotational latency of 3 milliseconds. The disk array is a RAID 5 disk array consisting of 8 Atlas 10K disks with a 32 KB stripe size. In both cases, the "real" device is provided by DiskSim [4], which has a validated model of the Atlas 10K disk. We replay all the traces on the two devices except the *SAP* trace, which is beyond the capacity of the Atlas 10K disk.

Traces. We use three sets of real-world traces in this study. Table 1 lists the summary statistics of the edited traces. The first two, *cello92* and *cello99* capture typical computer system research I/O workloads, collected at HP Labs in 1992 and 1999 respectively [12, 8]. We preprocess *cello92* to concatenate the LBNs of the three most active devices from the trace to fill the Atlas 10K disk. For *cello99*, we pick the three most active devices, among the 23 devices, and label them *cello99a*, *cello99b*, and *cello99c*. The *cello99* traces fit in a 9GB disk perfectly, so no trace editing is necessary. As these traces are long (two months for *cello92* and one year for *cello99*), we report data for a four-week snapshot (5/1/92 to 5/28/92 and 2/1/99 to 2/28/99).

The *SAP* trace was collected from an Oracle database server running SAP ISUCCS 2.5B in a power utility company. The server has more than 3,000 users and disk accesses reflect the retrieval of customer invoices for updating and reviewing. Sequential reads dominate the *SAP* trace.

Trace name	Length	Requests ($\times 10^6$)	Average Size	% of reads
cello92	4 weeks	7.8	12.9 KB	35.4%
cello99a	4 weeks	43.7	7.1 KB	20.9%
cello99b	4 weeks	13.9	118.0 KB	41.6%
cello99c	4 weeks	24.0	8.5 KB	26.4%
SAP	15 minutes	1.1	15.1 KB	99.9%

Table 1. Trace summary.

Evaluation methodology. The evaluation uses the device models to predict the average and 90th percentile response time for one-minute workload fragments. We report the median prediction errors over all the fragments using two metrics: absolute error defined as the difference between the predicted and the actual value, $|\hat{Y} - Y|$, and relative error defined as $\frac{|\hat{Y}-Y|}{Y}$.

We use the first two weeks of *cello99a* in training because of the trace's relatively rich access patterns. The training trace is 19,583 minutes long. Because of the large number of requests, we use a uniform sampling rate of 0.01 to reduce the number of requests to 218,942 in training the request-level model.

Predictors in comparison. We evaluate our two CART-based device models, denoted as `CART-request` and `CART-workload` in the remaining text, against three predictors.

- `constant` makes predictions using the average or quantile response time of the training trace.
- `periodic` divides a week into $24 \times 7 \times 60$ one-minute intervals and remembers the aggregate performance of the training workload for each interval. Prediction uses the corresponding value of the interval with the same offset within the week.
- `linear` does linear regression on the workload descriptions.

Note that the `constant` and `periodic` predictors model workloads rather than devices, because they do not take workload characteristics as input. Both predictors rely on the similarity between the training and testing workloads to produce accurate predictions. The difference between `linear` and `CART-workload`, on the other hand, shows the importance of using non-linear models, such as the CART models, in device modeling.

5.1. Calibrating Request-Level Models

This section describes how we select parameter values for k and l for the request-level device models.

Figure 3 shows the relative importance of the request description features in determining per-request response time

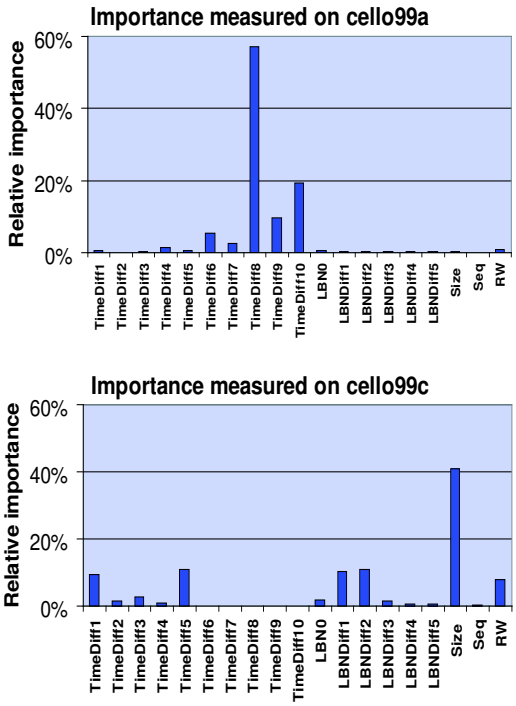


Figure 3. Relative importance of parameters in the request description for the Atlas 10K disk

by setting k to 10 and l to 5. The feature's relative importance is measured by its contribution in error reduction. The graphs show the importance of request description features measured on the Atlas 10K disk, trained on two traces (*cello99a* and *cello99c*). We use only the first day of the traces and reduce the data set size by 90% with uniform sampling.

We observe that the relative importance is workload dependent. As we expected, for busy traffic such as that which occurred in the *cello99a* trace, the queuing time dominates the response time, and thereby, the $TimeDiff$ features are more important. On the other hand, *cello99c* has small response times, and features that characterize the data transfer time, such as $Size$ and RW, have good predictive power in modeling the single disk.

We set k to 10 for $TimeDiff$ and l to 3 for $LBNDiff$ in the subsequent experiments so that we can model device behavior under both types of workloads.

5.2. Modeling A Single Disk

Figure 4 compares the accuracy of the five predictors in modeling an Atlas 10K 9GB disk on real-world traces.

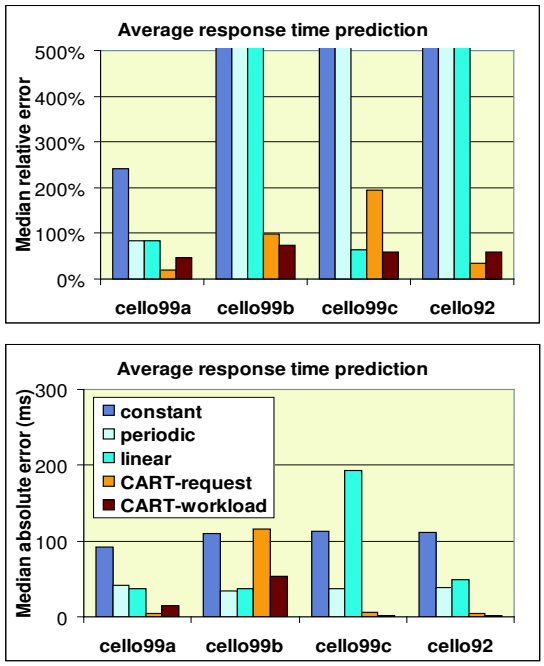


Figure 4. Comparison of predictors for a single 9GB Atlas 10K disk

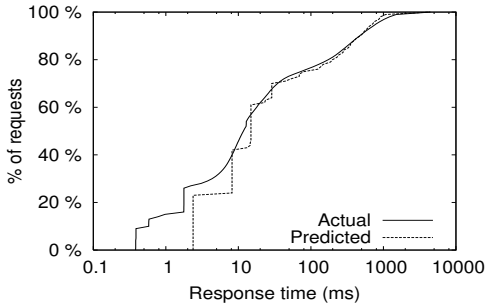


Figure 5. Cumulative response time distribution comparison.

More graphs can be found in [17]. As mentioned earlier, all predictors are trained using the first two weeks of *cello99a*. Overall, the two CART-based device models provide good prediction accuracy in predicting both the average and 90th percentile response times, compared to other predictors. Several more detailed observations can be made.

First, all of the models perform the best when the training and testing traces are from the same workload, i.e., *cello99a*, because the models have seen how the device behaves under such workloads. The `periodic` predictor also cuts the median prediction error of the `constant` predictor by more than a half because of the strong periodicity of the workload. `CART-request` and `CART-workload` further reduce the error to 4.84 milliseconds (19%) and 14.83 milliseconds (47%) respectively for the average response time prediction, and 20.46 milliseconds (15%) and 49.50 milliseconds (45%) respectively for the 90th percentile. The performance difference between `linear` and `CART-workload` roughly quantifies the benefit of using a non-linear model, such as CART, because both accept the same input. We observe a significant improvement from the former to the latter, suggesting non-linear device behavior.

Figure 5 further compares the predicted response time distribution by `CART-request` and the actual one. The model is built for the Atlas 10K disk. The training trace is the first day of *cello99a*, and the testing trace is the second day of the same trace. The actual and predicted average response times are 137 ms and 133 ms respectively. The corresponding demerit value, defined in [13] as the root mean square of horizontal distance between the actual and predicted curves in (b), is 46 milliseconds (33.4%). The long tail of the distribution is well captured by the request-level model, indicating that the request description is effective in capturing request-level characteristics needed to predict response times.

Second, both CART-based device models interpolate better across workloads than the other models. `constant` and `periodic` rely blindly on similarities between the training and testing workloads to make good predictions. Consequently, it is not surprising to see huge prediction errors when the training and testing workloads differ. The CART-based predictors, on the other hand, are able to distinguish between workloads of different characteristics and are more robust to the difference between the training and testing workloads.

Third, model accuracy is dependent on the training workload quality even for the CART-based models. The prediction error increases for workloads other than *cello99a*, because of the access pattern differences among these traces. The CART-based models learn device behavior through training; therefore, they cannot predict performance for workloads that have totally different characteristics from the training workloads. For example, `CART-request` constantly over-predicts for *cello99c*, because the model was never trained with the small sequential accesses that are particular to *cello99c*. Section 5.4 gives an informal error analysis and identifies inadequate training being the most significant error source.

Fourth, high quantile response times are more difficult to predict. We observe larger prediction errors from all the

predictors for 90th percentile response time predictions than for average response time predictions. The accuracy advantage of the two CART-based models is higher for 90th percentile predictions.

In summary, the two CART-based models give accurate predictions when the training and testing workloads share similar characteristics and interpolate well otherwise. The good accuracy suggests the effectiveness of the request and workload descriptions in capturing important workload characteristics.

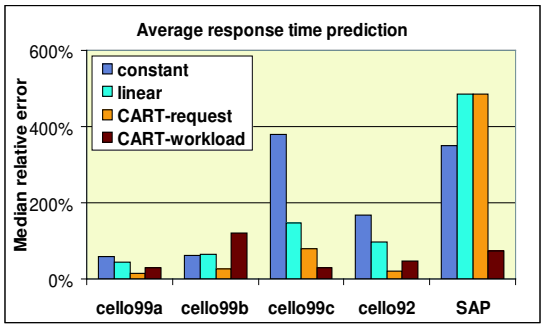


Figure 6. Comparison of predictors for a RAID 5 disk array of 8 Atlas 10K disks in predicting average response time.

5.3. Modeling A Disk Array

Figure 6 compares the accuracy of the four predictors in modeling the disk array. The `periodic` predictor is not presented because the *SAP* trace does not provide enough information on arrival time for us to know the offset within a week. The overall results are similar to those for the single disk. The two CART-based models are the most accurate predictors. The absolute errors become smaller due to the decreased response time from the single disk to the disk array. The relative accuracy among the predictors, however, stays the same. Overall, the CART-based device modeling approach works well for the disk array.

5.4. Error Analysis

This section presents an informal error analysis to identify the most significant error sources for the CART-based device models.

A model's error consists of two parts. The first part comes from intrinsic randomness of the input data, such as measurement error, and this error cannot be captured by any model. The rest of the error comes from the modeling approach itself. The CART-based models incur error at three

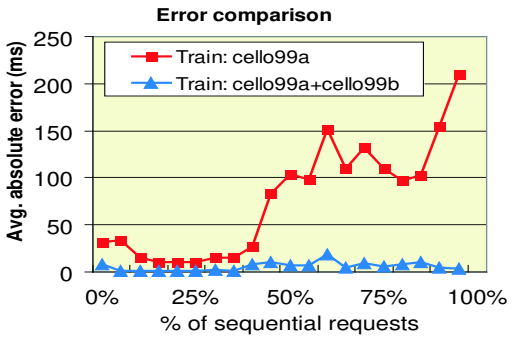


Figure 7. Effect of training workload.

places. First, the transformation from workloads to vectors introduces information loss. Second, the CART-based models use piece-wise constant functions, which could be different from the true functions. Third, a low-quality training trace yields inaccurate models because CART relies on the information from the training data to make predictions. An inadequate training set has only a limited range of workloads and leads to large prediction errors for workloads outside of this range. We find that the last error source, inadequate training data, causes the most trouble in our experiments.

We conduct a small experiment to verify our hypothesis. Figure 7 compares the prediction errors for workload fragments of different sequentiality. The spectrum of sequentiality (from 0% to 100% of requests in the workload being sequential) is divided into 20 buckets, and the graph shows the average absolute error for each bucket. When we train the device model using *cello99a*, the model incurs large errors for fragments of high sequentiality because *cello99g* has high-sequentiality fragments, while in *cello99a*, the sequentiality never goes beyond 0.5. The prediction error is reduced significantly when we include the first half of *cello99b* in training. The dramatic error reduction suggests that prediction errors from the other sources are negligible when compared with the ones introduced by inadequate training. We conclude from this evidence that contributing efforts in black-box device modeling should be directed toward generating a good training set that covers a broad range of workload types. Future research work can use synthetic workload generators to provide training traces of diverse characteristics.

6. Conclusions

Storage device performance modeling is an important element in self-managed storage systems and other application planning tasks. Our target model takes a workload as input and predicts its aggregate performance on the mod-

eled device efficiently and accurately. This paper presents our initial results in exploring machine learning tools to build device models. A black box predictive tool, CART, makes device models independent of the storage devices being modeled, and thus, general enough to handle any type of devices. This paper presents two ways of applying CART models, yielding the request-level and workload-level device models. Our experiments on real-world traces have shown that both types of models are accurate and efficient. The error analysis suggests that the quality of the training workloads plays a critical role in model accuracy. Continuing research can focus on improving the accuracy and efficiency of the models by investigating synthetic training workload generation and high-quality workload descriptions.

7. Acknowledgments

We thank the members and companies of the PDL Consortium (including EMC, Hewlett-Packard, Hitachi, Hitachi Global Storage Technologies, IBM, Intel, LSI Logic, Microsoft, Network Appliance, Oracle, Panasas, Seagate, Sun, and Veritas) for their interest, insights, feedback, and support. We thank IBM for partly funding this work through a CAS student fellowship and a faculty partnership award. This work is funded in part by NSF grants CCR-0205544, IIS-0133686, BES-0329549, IIS-0083148, IIS-0113089, IIS-0209107, and IIS-0205224. We would also like to thank Eno Thereska, Mike Mesnier, and John Strunk for their participation and discussion in the early stages of this project.

References

[1] E. Anderson, M. Kallahalla, S. Spence, R. Swaminathan, and Q. Wang. Ergastulum: Quickly fi nding near-optimal storage system designs. Technical Report HPL-SSP-2001-05, HP Labs, 2001.

[2] Eric Anderson. Simple table-based modeling of storage devices. Technical Report HPL-SSP-2001-4, HP Labs, 2001.

[3] L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[4] John Bucy, Greg Ganger, and contributors. The DiskSim simulation environment version 3.0 reference manual. Technical Report CMU-CS-03-102, Carnegie Mellon University, 2003.

[5] Shenze Chen and Don Towsley. A performance evaluation of RAID architectures. *IEEE Transactions on Computers*, 45(10):1116–1130, 1996.

[6] Gregory R. Ganger, John D. Strunk, and Andrew J. Klosterman. Self-* storage: Brick-based storage with automated administration. Technical Report CMU-CS-03-178, Carnegie Mellon University, 2003.

[7] María E. Gómez and Vicente Santonja. Analysis of self-similarity in I/O workload using structural modeling. In *7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 234–243, 1999.

[8] María E. Gómez and Vicente Santonja. A new approach in the modeling and generation of synthetic disk workload. In *Proceedings of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 199–206, 2000.

[9] International Business Machines Corp. Autonomic computing: IBM's perspective on the state of information technology. http://www.research.ibm.com/autonomic/manifesto/.

[10] Edward K. Lee and Randy H. Katz. An analytic performance model of disk arrays. In *Proceedings of the 1993 ACM SIGMETRICS*, pages 98–109, 1993.

[11] Arif Merchant and Guillermo A. Alvarez. Disk array models in Minerva. Technical Report HPL-2001-118, HP Laboratories, 2001.

[12] Chris Ruemmler and John Wilkes. Unix disk access patterns. In *Winter USENIX Conference*, pages 405–420, 1993.

[13] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, 27(3):17–28, 1994.

[14] Elizabeth Shriver, Arif Merchant, and John Wilkes. An analytical behavior model for disk drives with readahead caches and request reordering. In *Proceedings of International Conference on Measurement and Modeling of Computer Systems*, pages 182–191, 1998.

[15] Mustafa Uysal, Guillermo A. Alvarez, and Arif Merchant. A modular, analytical throughput model for modern disk arrays. In *Proceedings of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 183–192, 2001.

[16] Mengzhi Wang, Anastassia Ailamaki, and Christos Faloutsos. Capturing the spatio-temporal behavior of real traffi c data. *Performance Evaluation*, 49(1/4):147–163, 2002.

[17] Mengzhi Wang, Kinman Au, Anastassia Ailamaki, Anthony Brockwell, Christos Faloutsos, and Gregory R. Ganger. Storage device performance prediction with CART models. Technical Report CMU-PDL-04-103, Carnegie Mellon University, 2004.

[18] J. Wilkes. The Pantheon storage-system simulator. Technical Report HPL–SSP–95–14, Hewlett-Packard Laboratories, 1995.

[19] John Wilkes. Data services – from data to containers. Keynote address at File and Storage Technologies Conference (FAST'03), March – April 2003.